DTIC FILE COPY

2

AD-A223 333

CECOM

CENTER FOR SOFTWARE ENGINEERING

ADVANCED SOFTWARE TECHNOLOGY

cf. change - p. 19

Subject: **Final Report - Methodology Study for Real-Time Ada Problems**

CIN: **C02 091LA 0001**

24 MARCH 1989

90 002026

90 06 21 052

FINAL TECHNICAL REPORT

METHODOLOGY STUDY

FOR REAL-TIME ADA PROBLEMS

by

Sterling J. McCullough (Computer Technology Group, Ltd.)
Conrad S. Johnson (Sonicraft, Inc.)
Frederick C. Francl (Sonicraft, Inc.)

for

U.S. Army HQ CECOM
Center for Software Engineering
Advanced Software Technology
Fort Monmouth, NJ 07703-5000

16 FEBRUARY 1989

Contract No. DAAL03-D-0001
Delivery Order No. 0737
U.S. Army Research Office
Scientific Services Program

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF FIGURES

REAL-TIME ADA PROBLEMS METHODOLOGY STUDY

1. INTRODUCTION

1.1 Purpose

Current policy [DoD87] of the US Department of Defense (DoD) mandates that the Ada programming language (as well as validated Ada compilers and an Ada-based program design language [PDL]) be used on all integral weapon system computers. Generic problems that stem from this policy [Soni87], [LabT87] are the subject of this final technical report, which shows the relationship of each generic problem to the development methods (also called methodologies) used on real-time Ada projects.

1.2 Terminology

The definitions given in DoD Standard 2167A [DoD88] apply to all terms used in this report, with additional terms defined in ANSI/IEEE Std. 729-1983 [ANSI83] and in Appendix A of this report.

1.3 Organization

Section 1 explains the intent of the document.

Section 2 presents the approach used for the investigation as a whole, showing how the theoretical groundwork is integrated with the empirical portion. The initial findings are then presented and analyzed.

Section 3 summarizes the results of this technical effort, formulates the major conclusions, and recommends additional work.

KR

2. TECHNICAL DISCUSSION

Sonicraft used a three step approach in this investigation:

1) Analyze design methods currently in use for real-time Ada developments for their theoretical impact on generic Ada problems.

2) Interview participants of actual Ada projects to confirm or disprove the theoretical assertions about the impacts of design methods on generic real-time Ada problems.

3) Summarize the findings of the investigation in a matrix of generic real-time Ada problems vs. features of commonly-used design methods and in a data base that explains any impacts shown in the matrix.

2.1 Design Methods vs. Problems

Figure 1 presents the results of a theoretical analysis of the impact of design methods upon the generic real-time Ada problems. The figure shows that any of the problems impacted by one of the methods is also impacted by all the others, where impact is shown by an "X" at the intersection of the problem row with the method column.

Appendix C contains a printout from a database file which includes the generic Ada problem definition and the reasons why impact would be theoretically expected for each "X." Just as figure 1 showed redundancy across design methods for any problem, so too does Appendix C.

A further difficulty with looking at methods vs. problems was the diversity in the design methods in common use. A recent catalog of real-time design methods [Tele87] covered 41 established methods plus six emerging methods. The chances of finding mutually-reinforcing opinions on a method's effects are greatly diminished if very few respondents use the same methods.

Because of these difficulties, figure 2 was developed to replace figure 1. The rows in figure 2 show the same generic real-time Ada problems, but now the columns show potentially significant features of any design method. These features were used in the Appendix C theoretical explanations and were found to be much less redundant. This can also be seen in figure 2 by noting that very few features affect most Ada problems, and that no Ada problems are affected by at least half of the design method features.

The data in figure 2 was therefore taken as the theoretical baseline to be used in the succeeding steps in the study.

MATRIX OF PROBLEMS VS. METHODS (BASED ON THEORY)

PROB NO.	PROB TITLE	YOURDON	FUNCTIONAL DECOMPOSITION	OOD	SADT	PAMELA	WARNIER-ORR	JACKSON
1	LACK OF KNOWLEDGE CONCERNING ADA RTE							
2	IMPACT OF ADA COMP IMPL DIFFERENCES							
3	IMPACT OF INTRPT HNDLG O/H	X	X	X	X	X	X	X
4	IMPACT OF MEM MGMT O/H	X	X	X	X	X	X	X
5	IMPACT OF RSL O/H	X	X	X	X	X	X	X
6	IMPACT OF TASKING O/H	X	X	X	X	X	X	X
7	INEFF OF OBJ CODE GEN	X	X	X	X	X	X	X
8	RQMTS FOR EXTENSIVE ADA OPTIM	X	X	X	X	X	X	X
9	INADEQ CAP PROVIDED BY CURR DEBUG							
10	DELAYED HANDLING OF ADA EXCEPTIONS							
11	IMPACT OF EXTENSIVE USE OF GENERICS							
12	INABILITY TO PERF INDEP OF RSL							
13	LACK OF A DISTR RSL (MULTIPLE PROC)							
14	INABILTY TO PERF SECURE ADA PROC	X	X	X	X	X	X	X
15	DIVERSITY IN IMPL OF APSE'S	X	X	X	X	X	X	X
16	POOR PERFORMANCE OF ADA TOOLS							
17	DIFF IN BENCHMARKING ADA SYSTEMS	X	X	X	X	X	X	X
18	LACK OF ADA S/W DEVELOPMENT TOOLS	X	X	X	X	X	X	X
19	ADA LANGUAGE COMPLEXITY	X	X	X	X	X	X	X
20	RQMTS FOR CUSTOMIZATION OF RSL	X	X	X	X	X	X	X
21	LACK OF EXP ADA PROGRAMMERS	X	X	X	X	X	X	X
22	EXTENSIVE ADA TRNG REQUIREMENTS	X	X	X	X	X	X	X
23	INACCY OF C/S EST FOR ADA PROG	X	X	X	X	X	X	X
24	LACK OF ADA EST SW DEV METHOD	X	X	X	X	X	X	X
25	LACK OF EST ADA SW STDS AND GUIDE	X	X	X	X	X	X	X
26	PRODUCTIVITY IMPACTS OF ADA	X	X	X	X	X	X	X
27	IMPACT OF CONSTRAINT CHK ON SYS. PERF							
28	INABILITY TO ASSIGN DYN TASK PRIORITIES							
29	INABILITY TO PERF PARALLEL PROCESSING							
30	LACK OF SUPPT FOR LOW LEVEL OPERATION							
31	INABILITY TO PERF TASK RESTART							
32	INABILITY TO PERF CYCLIC SCH IN ADA							
33	LACK OF FLTING PT COPROCESSOR SUPPORT							
34	INABILITY TO RECOVER FROM CPU FAULTS							
35	IMPACT OF ADA ACVC ISSUES							
36	INABILITY TO PERF ASYNC. TASK							
37	LACK OF IMPLEMENTATION OF CHAPTER 13							

FIGURE 1

MATRIX OF PROBLEMS VS. METHOD FEATURES (BASED ON THEORY)

PROB NO.	PROB TITLE	PROCESS VISIBIL.	INFO HIDING	ADA ORIENTED	OVERUSE TASKING	METHODOL. TOOLS	EASE OF USE	EASE OF LEARNING	PROBLEM DEFINITION	DATA VISIBIL.	DESIGN QUALITY	TRACEABIL.	PROG/DATA STRUCTURE	PROCESS STATE	AUTO CODING	DESIGN CONSIST.
1	LACK OF KNOWLEDGE CONCERNING ADA RTE															
2	IMPACT OF ADA COMP IMPL DIFFERENCES															
3	IMPACT OF INTRPT HNDLG O/H	X														
4	IMPACT OF MEM MGMT O/H		X	X						X	X		X			
5	IMPACT OF RSL O/H	X														
6	IMPACT OF TASKING O/H	X		X	X									X		
7	INEFF OF OBJ CODE GEN			X	X						X				X	
8	RQMTS FOR EXTENSIVE ADA OPTIM			X	X						X			X	X	
9	INADEQ CAP PROVIDED BY CURR DEBUG															
10	DELAYED HANDLING OF ADA EXCEPTIONS															
11	IMPACT OF EXTENSIVE USE OF GENERICS															
12	INABILITY TO PERF INDEP OF RSL															
13	LACK OF A DISTR RSL (MULTIPLE PROC)															
14	INABILTY TO PERF SECURE ADA PROC		X	X						X	X		X			
15	DIVERSITY IN IMPL OF APSE'S					X					X					
16	POOR PERFORMANCE OF ADA TOOLS															
17	DIFF IN BENCHMARKING ADA SYSTEMS			X	X				X		X					X
18	LACK OF ADA S/W DEVELOPMENT TOOLS			X		X										
19	ADA LANGUAGE COMPLEXITY			X			X								X	
20	RQMTS FOR CUSTOMIZATION OF RSL			X	X				X		X					X
21	LACK OF EXP ADA PROGRAMMERS			X			X	X								
22	EXTENSIVE ADA TRNG REQUIREMENTS						X	X								
23	INACCY OF C/S EST FOR ADA PROG			X		X	X	X	X		X				X	X
24	LACK OF ADA EST SW DEV METHOD			X			X	X				X			X	
25	LACK OF EST ADA SW STDS AND GUIDE			X		X			X		X				X	X
26	PRODUCTIVITY IMPACTS OF ADA			X		X	X	X			X				X	X
27	IMPACT OF CONSTRAINT CHK ON SYS. PERF															
28	INABILITY TO ASSIGN DYN TASK PRIORITIES															
29	INABILITY TO PERF PARALLEL PROCESSING															
30	LACK OF SUPPT FOR LOW LEVEL OPERATION															
31	INABILITY TO PERF TASK RESTART															
32	INABILITY TO PERF CYCLIC SCH IN ADA															
33	LACK OF FLTING PT COPROCESSOR SUPPORT															
34	INABILITY TO RECOVER FROM CPU FAULTS															
35	IMPACT OF ADA ACVC ISSUES															
36	INABILITY TO PERF ASYNC. TASK															
37	LACK OF IMPLEMENTATION OF CHAPTER 13															

FIGURE [illegible]

2.2 Interviews

The theoretical assertions shown in figure 2 and described in Appendix C were tested by interviewing the selected project participants. The goal was to find what impact (if any) they thought the design method they used had upon each of the generic real-time Ada problems. This impact could be shown by alleviating the problem, perhaps to the extent that it didn't show up at all, or by aggravating it.

Before these interviews took place the participants were given a copy of the generic real-time Ada problem definitions [Soni 87], [LabT 87] to establish a common vocabulary.

2.3 Analysis of Responses

As was done for the theoretically expected impacts shown in figure 2, a matrix of empirically found impacts is shown in figure 3 for comparison. This matrix shows how many Ada projects found the impact to be as predicted by the theoretical baseline. No instances were found where the impact was the OPPOSITE of the theoretical prediction.

Just one report of impact is not significant to most observers. This has been formalized by defining statistical significance for this study:

> A confidence factor of 85 percent will be demanded before an assertion will be considered empirically established. Assuming a priority probability of 50% that a given assertion is true, and assuming a binomial distribution (i.e., the truth or falsity of an assertion on one project does not change the 50% probability of truth on any other project), this means that a minimum of three projects must report an assertion to be true. If even one project reports a disparate result, many additional projects must be interviewed to establish an assertion as being either true or false.

The explanations of each impact reported in figure 3 are given in Appendix D. For the sake of completeness, many of the explanations are given for impacts that do not meet the test above for statistical significance. None of the statistically insignificant impacts had any effect on the findings reported in paragraph 3.1 below.

MATRIX OF PROBLEMS VS. METHOD FEATURES (BASED ON ACTUAL INTERVIEWS)

PROB NO.	PROB TITLE	PROCESS VISIBIL.	INFO HIDING	ADA ORIENTED	OVERUSE TASKING	METHOD TOOLS	EASE OF USE	EASE OF LEARNING	PROBLEM DEFINITION	DATA VISIBIL.	DESIGN QUALITY	TRACEABIL.	PROG/DATA STRUCTURE	PROCESS STATE	AUTO CODING	DESIGN CONSIST.
1	LACK OF KNOWLEDGE CONCERNING ADA RTE															
2	IMPACT OF ADA COMP IMPL DIFFERENCES															
3	IMPACT OF INTRPT HNDLG O/H															
4	IMPACT OF MEM MGMT O/H															
5	IMPACT OF RSL O/H															
6	IMPACT OF TASKING O/H				(2)						(1)					
7	INEFF OF OBJ CODE GEN															
8	RQMNTS FOR EXTENSIVE ADA OPTIM															
9	INADEQ CAP PROVIDED BY CURR DEBUG															
10	DELAYED HANDLING OF ADA EXCEPTIONS															
11	IMPACT OF EXTENSIVE USE OF GENERICS															
12	INABILITY TO PERF INDEP OF RSL															
13	LACK OF A DISTR RSL (MULTIPLE PROC)															
14	INABILITY TO PERF SECURE ADA PROC		(1)											(1)		
15	DIVERSITY IN IMPL OF APSE'S															
16	POOR PERFORMANCE OF ADA TOOLS															
17	DIFF IN BENCHMARKING ADA SYSTEMS															
18	LACK OF ADA S/W DEVELOPMENT TOOLS					(6)										
19	ADA LANGUAGE COMPLEXITY															
20	RQMNTS FOR CUSTOMIZATION OF RSL															
21	LACK OF EXP ADA PROGRAMMERS			(3)		(1)	(6)	(3)								
22	EXTENSIVE ADA TRNG REQUIREMENTS					(1)	(8)	(7)								
23	INACCY OF C/S EST FOR ADA PROG															
24	LACK OF ADA EST SW DEV METHOD			(8)		(6)	(6)					(5)		(1)		(2)
25	LACK OF EST ADA SW STDS AND GUIDE			(9)			(2)									
26	PRODUCTIVITY IMPACTS OF ADA	(1)		(3)		(6)	(4)	(5)			(2)					
27	IMPACT OF CONSTRAINT CHK ON SYS. PERF															
28	INABILITY TO ASSIGN DYN TASK PRIORITIES															
29	INABILITY TO PERF PARALLEL PROCESSING															
30	LACK OF SUPPT FOR LOW LEVEL OPERATION															
31	INABILITY TO PERF TASK RESTART															
32	INABILITY TO PERF CYCLIC SCH IN ADA															
33	LACK OF FLTING PT COPROCESSOR SUPPORT															
34	INABILITY TO RECOVER FROM CPU FAULTS															
35	IMPACT OF ADA ACVC ISSUES															
36	INABILITY TO PERF ASYNC. TASK															
37	LACK OF IMPLEMENTATION OF CHAPTER 13															

FIGURE 3

2.4 Follow-up Interviews

A draft Final Technical Report was written to find exactly what still needed to be done. Having pulled together the interview results and having analyzed them, the biggest problem seemed to be the sparseness of impacts registered in figure 3 (as it was then).

Upon close examination of the detailed notes on each interview, It was determined that the biggest problem was that the project people interviewed wanted to focus on their particular design method rather than features of methods. This was a problem only because of the limited total time (usually an hour) which they were able to give us for the interview.

It is hard to criticize this tendency because the generic real-time Ada problems and specific design methods seem to be the natural relationship to study. It was only when the tremendous redundancies were found (paragraph 2.1) that our approach shifted to the emphasis on design method features instead of on the design methods themselves.

The immediate way to improve the situation within the scope of the present contract was to conduct short follow-up interviews with focus on the impacts that each particular Ada project SHOULD have seen. Since familiarity with the overall project had already been achieved (from the first interview) this kind of approach worked well.

3. SUMMARY OF RESULTS AND CONCLUSIONS

3.1 Results

The principal findings of this study were:

There was very strong agreement between the theoretical assessment (figure 2) and the empirical results (figure 3), with a few cases of relationships of problems to design methods that were not expected theoretically.

On the other hand, some interviewees had difficulty in assessing the impacts of method features on generic problems because of lack of familiarity with either the features or the problems.

Ada-oriented design methods, which were theoretically expected to help on more of the generic problems than any other method feature (see figure 2), had a large number of favorable impacts reported (see figure 3).

Good methodology tools also scored very highly, even though this was seemingly not that important a feature theoretically (see figure 2). The most popular tools were those that automated documentation or provided feedback on design quality.

Closely related to this was the strong preference toward tools and methodologies that are easy to learn and easy to use. As was previously reported [Soni87], the design methods are often much harder to master than the Ada programming language. An easier method therefore alleviated some of the problems associated with learning Ada.

The majority of the Ada projects surveyed did not use a single, recognized development method. Most used a hybrid of two or more common methods, and one project used distinctly different methods in each design phase and/or type of module being developed (difficult real-time constraints vs. data manipulations, for example).

3.2 Conclusions

Based on the findings of the study, Sonicraft reached two important conclusions:

The impact of design methods upon generic Ada real-time problems is limited to about half of the problem types. Many generic problems are highly dependent upon things like compiler features or performance, and these are virtually unaffected by method features. This was predicted theoretically (figure 2) and confirmed empirically (figure 3).

The manner of selecting the design method was surprising, however. We had expected some process analogous to compiler selection (features and performance comparisons, benchmarking, etc.), but found most developers just used one they already knew or that was recommended to them by upper management.

The surprising conclusion above stems from two study results given above. First, there is a general lack of familiarity with available methods and second, a large investment must be made to adopt a new design method.

10. APPENDIX A: DEFINITIONS

10. APPENDIX A: DEFINITIONS

Ada-oriented: The ability of a method to map the software design directly into the use of Ada language constructs such as packages, tasks, and generics.

Approach: A way of beginning or managing an effort; a way of analyzing, planning, or directing a project; a way of conducting operations. A scheme is an approach when it suggests ways to identify goals initially and/or suggests, at an abstract level, ways to proceed without goals. [Tele 87]

Auto Coding: Auto coding represents the capability, using a software tool, to automatically generate Ada source code statements. The tool user must provide specification information such as data definitions, module names, call decisions, etc.. [PJAC87]

Data Visibility: The extent to which a method provides visibility into the data that is used within an Ada software system. This visibility includes such issues as data flows, data definitions, and static vs. dynamic data.

Design Consistency: The extent to which an Ada software development method provides guidelines for consistent application of the method's principles. The benefit of this feature is that it encourages the development of a consistent software design among the different parts and participants of a software project.

Design Quality: The extent to which a method provides guidelines for determining and evaluating the design quality of Ada software programs. This feature also provides a measure of the quality of the software that is designed using a method (reliability, maintainability, testability, portability, correctness, efficiency, understandability, etc.)

Ease of Learning: The ease with which an Ada-oriented method can be learned. This includes factors such as the amount of training that is recommended (based on the complexity of a method), the levels of training that are recommended (beginner, intermediate, and advanced), and the amount of time that is required before a software engineer can effectively use a method.

Ease of Use: The ease with a method can be used to design and implement an Ada software system. Ease of use is based on the simplicity, completeness, and consistency of the underlying method's principles, terminology, symbology, and products (documentation and deliverables).

Efficiency: The extent to which a software component fulfills its purpose with minimum use of computer resources.

Information Hiding: The extent to which a method supports information hiding for Ada software systems (such as through the use of packages). The principle of information hiding suggests that modules should be specified and designed so that information (procedure and data) contained within a module is inaccessible to other modules that have no need for such information. [SEPA]

Method: A definite, established, logical, or systematic plan. The steps and purposes have been thought beforehand in detail. A scheme is a method when it guides the user to a predictable result, given an appropriate set of starting conditions. [Tele 87]

Methodology: The study of methods. [Tele 87]

Methodology Tools: A measure of the availability and maturity of automated software tools which implement an Ada-oriented software method. The issue of availability involves the variety of tools that exist, the hardware/software environments in which these tools run, and the interface between these tools and the APSE.

Overuse of Tasking: The extent to which a method uses tasks (in particular, the Ada tasking construct) to implement an Ada software design.

PAMELA: Process Abstraction Method for Embedded Large Applications, a trademark of George Cherry for his Ada design method.

Portability: The ease with which a software component can be made functional in a different application or target computer architecture.

Problem Definition: The extent to which a method allows a software developer to define and represent a problem and its proposed solution during the development of an Ada software system. This is based on the completeness of the underlying principles, terminology, and symbology of a method. It is a measure of the ability of a method to model and represent the real-world.

Process State: The extent to which a method provides information concerning the state of the various processes which make up an Ada software system. This includes guidelines to establish state changes for Ada design efforts and guidelines for using symbology to represent state changes.

Process Visibility: The extent to which a method provides visibility into the functions and interfaces of the processes within an Ada software system. This visibility includes such issues as process control flow, concurrency (tasks), and external control (interrupts).

Program and Data Structure: The extent to which a method provides guidelines for using Ada language features which impose program and data structure. The program/data structure can take a number of forms, to include a flat hierarchy or a layered hierarchy.

Real-Time Ada: A computer program written in the Ada language which implements one or more real-time functions, usually triggered by interrupts.

Real-Time Function: Any system function (hardware, software or a combination) which is considered to have faulted if it has not been completed within a specified time after a signal to start.

Reusability: The ease with which a software component can be instantiated for another application.

Runtime Support or Runtime Support Library (RSL): The set of embedded firmware required to interface the application's object code generated by the compiler to the target machine instruction set.

Task: Any program unit which is designed to be able to operate in parallel with other program units and to synchronize with them where necessary.

Traceability: The extent to which an Ada-oriented software development method provides the ability to validate the products of the various steps of a method and to verify that the input to each step fulfills the requirements levied by the previous step.

20. APPENDIX B: REFERENCES

20. APPENDIX B: REFERENCES

[ANSI83] ANSI/IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terms.

[Boeh81] B.Boehm, "Software Engineering Economics", Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Boyd86] S.Boyd,"Ada Methods: Object-Oriented Design & PAMELA", SIGAda, November 1986.

[Broo75] F.P.Brooks,"The Mythical Man-Month", 1975, Addison-Wesley, Reading, Mass.

[Broo87] F.P.Brooks,"No Silver Bullet", IEEE Computer, April 87.

[Burg87] T.M. Burger and K.W. Nielsen, "An Assessment of the Overhead Associated with Tasking Facilities and Task Paradigms in ADA", Hughes Aircraft Company, 1987.

[Davi85] H.Davis, "Measuring the Programmer's Productivity", Engineering Manager, February 1985.[Davi85].

[DoD87] Department of Defense Directive 3405.2, SUBJECT: Use of Ada in Weapon Systems, March 30, 1987.

[DoD88] DOD-STD-2167A, Defense System Software Development, 29 Februray 1988.

[Doty78] D.L.Doty, P.J.Nelson, and K.R.Stewart,"Software Cost Estimation Study: Guidelines for Improved Software Cost Estimating" (Vol 2), Final Technical Report by Doty Associates, Inc., for Rome Air Development Center (RADC-TR-77-220),Griffiss Air Force Base, NY, August 1977, 145pp as cited by: W. Myers,"A Statistical Approach to Scheduling Software Development", IEEE Computer, December 1978.

[Hans85] S.J.Hanson and R.R.Rosinski,"Programmer Perceptions of Productivity and Programming Tools", Communications of the ACM, February 1985.

[Jens85] R.W. Jensen, "Projected Productivity Impact of Near-Term Ada Use in Software System Development, "Hughes Aircraft Co., Fullerton, CA 1985.

[Keme87] C.Kemerer, "An Empirical Validation of Software Cost Estimating Models", Communications of the ACM, May 1987.

[LabT87] LabTek Corporation, "Software Engineering Issues on Ada Technology Insertion For Real-Time Embedded Systems," Final Technical Report to Center for Software Engineering, US Army CECOM, 24 July 1987.

[PCRA86] Policy Committee Reports From Armed Services, SIGAda, November 1986.

[PJAC87] "Proceedings of Joint Ada Conference - 1987", U.S. Army CECOM Pg. 229.

[SEPA] "Software Engineering: A Practitioner's Approach", McGraw-Hill, Pg.156.

[Soni87] Sonicraft, Inc.,"Software Engineering Problems Using Ada in Computers Integral to Weapon Systems," Final Technical Report to Center for Software Engineering, US Army CECOM, 9 October 1987.

[Soni88] Sonicraft, Inc., "Ada Methodology Study", Final Technical Report to Center for Software Engineering, US Army CECOM, 17 February 1988.

[Tele87] Teledyne Brown Engineering, Software Methodology Catalog, Final Technical Report to Center for Software Engineering, US Army CECOM, October 1987.

30. APPENDIX C: METHODS VS. PROBLEMS

30.3 IMPACT OF INTERRUPT HANDLING OVERHEAD ON SYSTEM PERFORMANCE

With ~~the embedded~~ all systems ~~being~~ now the ~~primary~~ overall target for the use of the Ada language, the efficient handling of interrupts becomes a major issue. Interrupts can be defined as hardware or software signals that stop the current processes of the system under specified conditions and in such a way that the processes can be resumed. In the embedded system environment, interrupts are critical to the ability of the system to respond to real-time events and perform its required functions. Interrupts, in general, signal the occurrence of some predefined event to the embedded system. The embedded system must then perform the correct functions in some determined amount of time. Therefore, any overhead time, time spent on processes that are over and above the required processes, will degrade the ability of the embedded system to meet its functional requirements. To better illustrate the problem, please note the following:

```
|<---------- B ------------->|<------------- C ------------->|
A
```

A : Represents the occurrence of a interrupt.

B : Represents the overhead time required to stop the current process and switch to the interrupt handler process.

C : Represents the time spent performing the required processes in response to the interrupt.

The problem that exists with Ada today is the overhead time "B" that is required in a Ada program to stop and switch to the interrupt handling process is too large and in many cases unacceptable for embedded system applications. Currently, Ada programs have overhead times in the hundreds of microseconds to milliseconds. The non-Ada embedded system application overhead times have been in tens of microseconds or less.

The following methodology features affect this generic Ada problem (Problem #3):

- o Process visibility

30.3.1 Yourdon

The Yourdon methodology affects the "Impact of Interrupt Handling Overhead" problem in the following ways:

Process visibility: The Yourdon methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the interrupt handling overhead on system performance and to modify the design accordingly. This can reduce the impact of interrupt handling overhead.

30.3.2 Functional Decomposition

The Functional Decomposition methodology affects the "Impact of Interrupt Handling Overhead" problem in the following ways:

Process visibility: The Functional Decomposition methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the interrupt handling overhead on system performnace and to modify the design accordingly. This can reduce the impact of interrupt handling overhead.

30.3.3 Object Oriented Development

The OOD methodology affects the "Impact of Interrupt Handling Overhead" problem in the following ways:

Process visibility: The OOD methodology addresses timing, external control, and interfaces to the operating system, and concurrency. This provides a means to evaluate the impact of the interrupt handling overhead on system performance and to modify the design accordingly. This can reduce the impact of interrupt handling overhead.

30.3.4 Structured Analysis and Design Technique

The SADT methodology affects the "Impact of Interrupt Handling Overhead" problem in the following ways:

Process visibility: The SADT methodology addresses timing, external control, and interfaces to the operating system, and concurrency. This provides a means to evaluate the impact of the interrupt handling overhead on system performance and to modify the design accordingly. This can reduce the impact of interrupt handling overhead.

30.3.5 PAMELA

The PAMELA methodology affects the "Impact of Interrupt Handling Overhead" problem in the following ways:

Process visibility: The PAMELA methodology addresses timing, external control, and interfaces to the operating system, and concurrency. This provides a means to evaluate the impact of the interrupt handling overhead on system performance and to modify the design accordingly. This can reduce the impact of interrupt handling overhead.

30.3.6 Warnier-Orr

The Warnier-Orr methodology affects the "Impact of Interrupt Handling Overhead" problem in the following ways:

Process visibility: The Warnier-Orr methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the interrupt handling overhead on system performance and to modify the design accordingly. This can reduce the impact of interrupt handling overhead.

30.3.7 Jackson

The Jackson methodology affects the "Impact of Interrupt Handling Overhead" problem in the following ways:

Process visibility: The Jackson methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the interrupt handling overhead on system performance and to modify the design accordingly. This can reduce the impact of interrupt handling overhead.

30.4 IMPACT OF MEMORY MANAGEMENT OVERHEAD

The run-time support provided to Ada applications programs by the Ada RSL includes a number of memory management functions. The primary functions are memory allocation and deallocation and heap storage management (garbage collection). These functions are performed by the RSL either on an as needed basis (memory allocation/deallocation during a context switch) or as required by the application (use of access types to create objects at run-time).

However, the RSL memory management features add a significant amount of run-time overhead to the performance of an Ada system. Since these functions are resident in the target machine and operate in conjunction with the application programs, they also require system resources (CPU,memory). The utilization of system resources by the RSL must be addressed when performing overall system sizing and timing analyses in the applications environment. It is important to know whether the overhead associated with the RSL memory management features is large enough to affect the ability of the system to meet specified requirements.

The following methodology features affect this generic Ada problem (Problem #4):

- Information hiding
- Ada-oriented
- Data visibility
- Design quality
- Program/data structure

30.4.1 Yourdon

The Yourdon methodology affects the "Impact of Memory Management Overhead" problem in the following ways:

Information hiding: The Yourdon methodology encourages the localization and logical/physical grouping of data, thus producing a design which requires reduced internal memory management. This reduces the impact of Ada memory management overhead.

Data visibility: The Yourdon methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This reduces the impact of Ada memory management overhead.

Design quality: The Yourdon methodology provides strong guidelines for determining the quality of the design in the areas of modularity and localization and tends to produce a design which has loose data coupling. The loose data coupling requires less internal memory management. This reduces the impact of Ada memory management overhead.

Program/data structure: The Yourdon methodology tends to result in sequentially cohesive program and data structures which reduce the amount of data shared between programs. This reduces the impact of Ada memory management overhead.

30.4.2 Functional Decomposition

The Functional Decomposition methodology affects the "Impact of Memory Management Overhead" problem in the following ways:

Information hiding: The Functional Decomposition methodology encourages the localization of data, thus producing a design which requires reduced internal memory management. This reduces the impact of Ada memory management overhead.

Data visibility: The Functional Decomposition methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This reduces the impact of Ada memory management overhead.

Design quality: The Functional Decomposition methodology tends to produce a design which has moderately tight data coupling, thus requiring increased internal memory management. This increases the impact of Ada memory management overhead.

Program/data structure: The Functional Decomposition methodology tends to result in program and data structures with poor cohesion, which increases the amount of data shared between programs. This increases the impact of Ada memory management overhead.

30.4.3 Object Oriented Development

The OOD methodology affects the "Impact of Memory Management Overhead" problem in the following ways:

Information hiding: The OOD methodology encourages the localization and logical/physical grouping of data through the use of Ada packages, thus producing a design which requires reduced internal memory management. This reduces the impact of Ada memory management overhead.

Ada-oriented: The OOD methodology leads directly to the use of Ada packages to encapsulate objects and the functions that operate on them, which leads to programs that are loosely data coupled. The loose data coupling requires less internal memory management. This reduces the impact of Ada memory management overhead.

Data visibility: The OOD methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This reduces the impact of Ada memory management overhead.

Design quality: The OOD methodology tends to produce a design which has loose data coupling, due in large part to the concept of packaging objects and their related functions together. The loose data coupling requires less internal memory management. This reduces the impact of Ada memory management overhead.

Program/data structure: The OOD methodology tends to result in programs that are communicationally cohesive (elements related by reference to same data) and packaged data structures which reduce the amount of data shared between programs. The loose data coupling requires less internal memory management. This reduces the impact of Ada memory management overhead.

30.4.4 Structured Analysis and Design Technique

The SADT methodology affects the "Impact of Memory Management Overhead" problem in the following ways:

Information hiding: The SADT methodology encourages the localization and logical/physical grouping of data, thus producing a design which requires reduced internal memory management. This reduces the impact of Ada memory management overhead.

Data visibility: The SADT methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This reduces the impact of Ada memory management overhead.

Design quality: The SADT methodology provides the capability to model the system data (inputs, outputs) and can produce a design which has loose data coupling. The loose data coupling requires less internal memory management. This reduces the impact of Ada memory management overhead.

30.4.5 PAMELA

The PAMELA methodology affects the "Impact of Memory Management Overhead" problem in the following ways:

Information hiding: The PAMELA methodology encourages the localization and logical/physical grouping of data through the use of Ada packages, thus producing a design which requires reduced internal memory management. This reduces the impact of Ada memory management overhead.

Ada-oriented: The PAMELA methodology leads directly to the use of Ada packages to encapsulate objects and the functions that operate on them, which leads to programs that are loosely data coupled. The loose data coupling requires less internal memory management. This reduces the impact of Ada memory management overhead.

Data visibility: The PAMELÁ methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This reduces the impact of Ada memory management overhead.

Design quality: The PAMELA methodology tends to produce a design which has loose data coupling, due in large part to the concept of packaging objects and their related functions together. The loose data coupling requires less internal memory management. This reduces the impact of Ada memory management overhead.

Program/data structure: The PAMELA methodology tends to result in programs that are sequentially cohesive (elements are partitions of the system data flow diagram) and packaged data structures which reduce the amount of data shared between programs. The loose data coupling requires less internal memory management. This reduces the impact of Ada memory management overhead.

30.4.4 **Warnier-Orr**

The Warnier-Orr methodology affects the "Impact of Memory Management Overhead" problem in the following ways:

Information hiding: The Warnier-Orr methodology encourages the localization and logical/physical grouping of data, thus producing a design which requires reduced internal memory management. This reduces the impact of Ada memory management overhead.

Data visibility: The Warnier-Orr methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This reduces the impact of Ada memory management overhead.

Design quality: The Warnier-Orr methodology tends to produce a design which has good localization of data. However, the program structure is created by mapping the programs into the data structure, which can result in low cohesion and requires more internal memory management. This increases the impact of Ada memory management overhead.

Program/data structure: The Warnier-Orr methodology tends to result in program structures with fair-to-poor cohesion, which increases the amount of data shared between programs. This increases the impact of Ada memory management overhead.

30.4.7 **Jackson**

The Jackson methodology affects the "Impact of Memory Management Overhead" problem in the following ways:

Information hiding: The Jackson methodology encourages the localization and logical/physical grouping of data, thus producing a design which requires reduced internal memory management. This reduces the impact of Ada memory management overhead.

Data visibility: The Jackson methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This reduces the impact of Ada memory management overhead.

Design quality: The Jackson methodology tends to produce a design which has good localization of data. However, the program structure is created by mirroring the data structure, which can result in low cohesion and requires more internal memory management. This increases the impact of Ada memory management overhead.

Program/data structure: The Jackson methodology tends to result in program structures with fair-to-poor cohesion, which increases the amount of data shared between programs. This increases the impact of Ada memory management overhead.

30.5 IMPACT OF RUN-TIME SUPPORT LIBRARY OVERHEAD ON SYSTEM PERFORMANCE

The Runtime Support Library (RSL) is the total package of software required at run-time to support the execution of the object code generated by the compiler for the application program. Functions such as dynamic memory management, system activation/allocation, interrupt processing, input/output operations, co-processor support, and tasking are all performed by the RSL. The basic problem with the RSLs of today is that they are generally too large and too slow for many embedded system applications. In terms of sizing, the problem is more noticeable when the application program is fairly small. In this case,it's very possible that the RSL can be larger than the application program. As the application programs grow larger, the proportion of memory taken by the RSL become less, thus the impact is not as great. For timing impacts, the amount of overhead experienced will depend upon what features of the language are used. Generally speaking, the more you use the unique features (tasking, dynamic memory, delays, etc.) of Ada, the more overhead, that is incurred. This can be attributed in part to the immaturity of existing RSLs. Because some of the Ada features are new to the implementors of the language, the techniques of implementing them efficiently are still emerging.

As an example of what some contractors are experiencing in RSL overhead, on Sonicraft's MEECN (Minimum Essential Emergency Communications Network) project it was discovered that the RSL alone would require over ninety kilobytes (KB). Of course, optimization efforts had to begin immediately to reduce size since the system was limited in its memory and power usage.

The following methodology features affect this generic Ada problem (Problem #5):

- o Process visibility

30.5.1 Yourdon

The Yourdon methodology affects the "Impact of RSL Overhead" problem in the following ways:

Process visibility: The Yourdon methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the RSL overhead on system performance and to modify the design accordingly. This reduces the impact of RSL overhead.

30.5.2 Functional Decomposition

The Functional Decomposition methodology affects the "Impact of RSL Overhead" problem in the following ways:

Process visibility: The Functional Decomposition methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the RSL overhead on system performance and to modify the design accordingly. This reduces the impact of RSL overhead.

30.5.3 Object Oriented Development

The OOD methodology affects the "Impact of RSL Overhead" problem in the following ways:

Process visibility: The OOD methodology addresses timing, external control, interfaces to the operating system, and concurrency. This provides a means to evaluate the impact of the RSL overhead on system performance and to modify the design accordingly. This reduces the impact of RSL overhead.

30.5.4 Structured Analysis and Design Technique

The SADT methodology affects the "Impact of RSL Overhead" problem in the following ways:

Process visibility: The SADT methodology addresses timing, external control, and interfaces to the operating system, and concurrency. This provides a means to evaluate the impact of the RSL overhead on system performance and to modify the design accordingly. This reduces the impact of RSL overhead.

30.5.5 PAMELA

The PAMELA methodology affects the "Impact of RSL Overhead" problem in the following ways:

Process visibility: The PAMELA methodology addresses timing, external control, interfaces to the operating system, and concurrency. This provides a means to evaluate the impact of the RSL overhead on system performance and to modify the design accordingly. This reduces the impact of RSL overhead.

30.5.6 Warnier-Orr

The Warnier-Orr methodology affects the "Impact of RSL Overhead" problem in the following ways:

Process visibility: The Warnier-Orr methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the RSL overhead on system performance and to modify the design accordingly. This reduces the impact of RSL overhead.

30.5.7 Jackson

The Jackson methodology affects the "Impact of RSL Overhead" problem in the following ways:

Process visibility: The Jackson methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the RSL overhead on system performance and to modify the design accordingly. This reduces the impact of RSL overhead.

30.6 IMPACT OF TASKING OVERHEAD ON SYSTEM PERFORMANCE

One of the key features of the Ada language is tasking. Ada tasks are "entities whose executions proceed in parallel [Brug87]."This feature gives Ada a great advantage or other high-level languages, but not without a price. The cost is in terms of overhead. Tasking overhead affects the efficiency of the system in both sizing and timing.

Whenever a designer decides to utilize tasking in an Ada program, he will automatically incur an additional cost in terms of additional run-time support code, which can be as high as thirty kilobytes. This code is required to perform the various features (entry calls, accepts, selects,..etc.) of Ada tasking at run-time. Another sizing problem has to do with the stack requirements of tasks. The designer must allocate enough memory for his application to make available the additional stack memory for task control information. Also, any stack memory required for any run-time procedures called to execute a particular feature must be added to the total size of the task stack allocation. The stack allocation requirements are required for each task declared in the designer's application program. Thus, the problem is compounded.

With the use of tasking, today's applications will experience timing overhead impacts due to tasking features like task allocation, task activation/termination, task switching, synchronization and task rendezvous. To determine what kind of overhead would be incurred by using tasking, a study was performed by Hughes Aircraft Company. The study conducted a series of tests using the DEC Ada Compiler (1.2) on a VAX 8600 (VMS 4.2). The following results show the magnitude of task overhead compared to the processing done within the task itself. [Brug87]:

Description	Task Overhead (usec)	Normal Proc. (usec)
1. Task activation and termination	1960	178
2. Task created via an allocator	150	14
3. Producer-Consumer (2 task switches)	503	46
4. Producer-Buffer-Consumer	1220	111
5. Producer-Buffer-Transporter-Consumer	1694	154
6. Producer-Transpt-Buffer-Transpt-Consumer	2248	204
7. Relay	906	82
8. Conditional Entry		
- no rendezvous	170	15
- rendezvous	29	3
9. Timed Entry		
- no rendezvous	254	23
- with rendezvous	33	3
10. Selective Wait with Terminate	127	12
11. Exception during a rendezvous	962	87

The following methodology features affect this generic Ada problem (Problem #6):

- o Process visibility
- o Ada-oriented
- o Overuse of tasking
- o Process state

30.6.1 Yourdon

The Yourdon methodology affects the "Impact of Tasking Overhead" problem in the following ways:

Process visibility: The Yourdon methodology addresses timing, external control, concurrency, and interfaces to the operating system. This provides a means to evaluate the impact of the tasking overhead on system performance and to modify the design accordingly. This can reduce the impact of tasking overhead.

30.6.2 Functional Decomposition

The Functional Decomposition methodology affects the "Impact of Tasking Overhead" problem in the following ways:

Process visibility: The Functional Decomposition methodology addresses timing, external control, concurrency, and interfaces to the operating system. This provides a means to evaluate the impact of the tasking overhead on system performance and to modify the design accordingly. This can reduce the impact of tasking overhead.

30.6.3 Object Oriented Development

The OOD methodology affects the "Impact of Tasking Overhead" problem in the following ways:

Process visibility: The OOD methodology addresses timing, external control, concurrency, and interfaces to the operating system. This provides a means to evaluate the impact of the tasking overhead on system performance and to modify the design accordingly. This can reduce the impact of tasking overhead.

Ada-oriented: The OOD methodology directly encourages the use of Ada tasks. OOD provides good guidance in the use of tasks to design Ada programs and leads to a more efficient use of tasks than a non-Ada-oriented methodology. This can reduce the impact of tasking overhead.

Overuse of tasking: The OOD methodology directly uses the Ada task construct to address concurrent processing, interrupt handling, management of shared resources, and other functions. OOD strives to control system complexity by minimizing the number of tasks that are required to perform processing. This can reduce the impact of tasking overhead.

30.6.4 Structured Analysis and Design Technique

The SADT methodology affects the "Impact of Tasking Overhead" problem in the following ways:

Process visibility: The SADT methodology addresses timing, external control, concurrency, and interfaces to the operating system. This provides a means to evaluate the impact of the tasking overhead on system performance and to modify the design accordingly. This can reduce the impact of tasking overhead.

Process state: The SADT methodology provides the developer with information concerning process states. This information is useful in evaluating the impact of the proposed tasking approach to minimize the overhead. This can reduce the impact of tasking overhead.

30.6.5 PAMELA

The PAMELA methodology affects the "Impact of Tasking Overhead" problem in the following ways:

Process visibility: The PAMELA methodology addresses timing, external control, concurrency, and interfaces to the operating system. This provides a means to evaluate the impact of the tasking overhead on system performance and to modify the design accordingly. This can reduce the impact of tasking overhead.

Ada-oriented: The PAMELA methodology directly encourages the use of Ada tasks. PAMELA provides good guidance in the use of tasks to design Ada programs and leads to a more effective use of tasks than a non-Ada-oriented methodology. This can reduce the impact of tasking overhead.

Overuse of tasking: The PAMELA methodology directly uses the Ada task construct to address concurrent processing, interrupt handling, management of shared resources, and other functions. PAMELA heavily emphasizes the use of concurrency to perform its processing; thus, the design uses a large number of tasks. This can greatly increase the impact of tasking overhead.

Process state: The PAMELA methodology provides the developer with information concerning process states. This information is useful in evaluating the impact of the proposed tasking approach to minimize the overhead. This can reduce the impact of tasking overhead.

30.6.6 Warnier-Orr

The Warnier-Orr methodology affects the "Impact of Tasking Overhead" problem in the following ways:

Process visibility: The Warnier-Orr methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the tasking overhead on system performance and to modify the design accordingly. This can reduce the impact of tasking overhead.

30.6.7 Jackson

The Jackson methodology affects the "Impact of Tasking Overhead" problem in the following ways:

Process visibility: The Jackson methodology addresses timing, external control, and interfaces to the operating system. This provides a means to evaluate the impact of the tasking overhead on system performance and to modify the design accordingly. This can reduce the impact of tasking overhead.

30.7 INEFFICIENCY OF OBJECT CODE GENERATED BY ADA COMPILERS

As with most first generation compilers for new languages, the Ada compilers today are somewhat inefficient because of the complexity of the Ada language. Unfortunately, the lack of efficient compilers directly impacts the development of embedded systems today. Embedded systems typically have very restrictive requirements on sizing, timing and power consumption. Therefore, any inefficiencies in the object code generation will impact the cost and performance of the typical weapon system. Because the compilers are producing more code than required to implement a particular function the compilation time is longer. As a result, it takes developers somewhat longer to complete the coding process. This means schedule and cost impacts.

The following methodology features affect this generic Ada problem (Problem #7):

- o Ada oriented
- o Overuse of tasking
- o Design quality
- o Auto coding

30.7.1 Yourdon

The Yourdon methodology affects the "Inefficiency of Object Code Generation" problem in the following ways:

Design quality: The Yourdon methodology tends to produce designs which require a large amount of parameter passing between modules. It also tends to produce deep program hierarchies. These factors can increase the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This can reduce the efficiency of generated object code.

30.7.2 Functional Decomposition

The Functional Decomposition methodology affects the "Inefficiency of Object Code Generation" problem in the following ways:

Design quality: The Functional Decomposition methodology tends to produce designs which have fair to poor cohesion and are somewhat tightly coupled. These factors can increase the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This can reduce the efficiency of generated object code.

30.7.3 Object Oriented Development

The OOD methodology affects the "Inefficiency of Object Code Generation" problem in the following ways:

Ada-oriented: The OOD methodology directly encourages the use of Ada constructs such as packages, tasks, and generics. OOD provides good guidance in the use of these constructs to design Ada programs and leads to a more efficient use of these features than a non-Ada-oriented methodology. This can increase the efficiency of generated object code.

Overuse of tasking: The OOD methodology directly uses the Ada task construct to address concurrent processing, interrupt handling, management of shared resources, and other functions. OOD strives to control system complexity by minimizing the number of tasks that are required to perform processing. This can increase the efficiency of generated object code.

Design quality: The OOD methodology tends to produce designs which have loose data coupling (due to packaging) and which are communicationally cohesive (functions are related by references to same data). These factors can decrease the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This increases the efficiency of generated object code.

30.7.4 Structured Analysis and Design Technique

The SADT methodology affects the "Inefficiency of Object Code Generation" problem in the following ways:

Design quality: The SADT methodology tends to produce designs which have loose data coupling and which are sequentially cohesive (elements are partitions of system data flow diagram). These factors can decrease the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This increases the efficiency of generated object code.

30.7.5 PAMELA

The PAMELA methodology affects the "Inefficiency of Object Code Generation" problem in the following ways:

Ada-oriented: The PAMELA methodology directly encourages the use of Ada constructs such as packages, tasks, and generics. PAMELA provides good guidance in the use of these constructs to design Ada programs and leads to a more efficient use of these features than a non-Ada-oriented methodology. This can increase the efficiency of generated object code.

Overuse of tasking: The PAMELA methodology directly uses the Ada task construct to address concurrent processing, interrupt handling, management of shared resources, and other functions. PAMELA heavily emphasizes the use of concurrency to perform its processing; thus, the design uses a large number of tasks. This can greatly increase the inefficiency of generated object code.

Design quality: The PAMELA methodology tends to produce designs which have loose data coupling (due to packaging) and which are sequentially cohesive (elements are partitions of the system data flow diagram). These factors can decrease the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This increases the efficiency of generated object code.

Auto coding: The PAMELA methodology allows the developer to automatically generate Ada source code based on the program design specifications. As a rule, code that is automatically generated is less efficient than that of a hand-coded program. This could reduce the efficiency of generated object code.

30.7.6 Warnier-Orr

The Warnier-Orr methodology affects the "Inefficiency of Object Code Generation" problem in the following ways:

Design quality: The Warnier-Orr methodology tends to produce designs which have low cohesion. Also, the data-oriented approach used in this methodology is not particularly suited for use in developing real-time (process-oriented) systems. These factors can increase the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This can reduce the efficiency of generated object code.

30.7.7 Jackson

The Jackson methodology affects the "Inefficiency of Object Code Generation" problem in the following ways:

Design quality: The Jackson methodology tends to produce designs which have low cohesion. Also, the data-oriented approach used in this methodology is not particularly suited for use in developing real-time (process-oriented) systems. The "serial file" concept of the Jackson methodology introduces inefficiency by creating intermediate program steps for data format conversions between input and output. These factors can increase the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This can greatly reduce the efficiency of generated object code.

30.8 NEED FOR EXTENSIVE ADA OPTIMIZATION

Due to the inefficiency of compiler-generated Ada applications code and the excessive overhead associated with the RSL run-time support, extensive optimization is generally required to improve the performance of Ada systems.

The types of system optimization that are performed include:

* Customizing the RSL for a particular application
* Reducing RSL overhead (tasking, memory management, interrupts)
* Adding special-purpose hardware and software
* Rewriting applications programs (algorithms)
* Modifying compiler implementation details
* Using non-Ada software
* Providing absolute addressing capability
* Providing for storage of constants in ROM

However, this extensive optimization can adversely affect system performance and project productivity. The addition of special-purpose hardware and software along with the use of project-specific programs can reduce system portability, reliability, maintainability, reusability, and verifiability, and can increase system complexity. Also, the extensive rework that is performed as part of the optimization efforts can decrease overall project productivity.

The following methodology features affect this generic Ada problem (Problem #8):

- o Ada-oriented
- o Overuse of tasking
- o Design quality
- o Process State
- o Auto coding

30.8.1 Yourdon

The Yourdon methodology affects the "Requirements for Extensive Ada Optimization" problem in the following ways:

Design quality: The Yourdon methodology tends to produce an inefficient Ada design due to extensive parameter passing and deep program hierarchies, which can cause increased Ada context switching and reduce system performance. This can increase the requirement to perform Ada optimization.

30.8.2 Functional Decomposition

The Functional Decomposition methodology affects the "Requirements for Extensive Ada Optimization" problem in the following ways:

Design quality: The Functional Decomposition methodology tends to produce an inefficient Ada design due to fair-to-poor program cohesion and somewhat tight data coupling, which can cause increased context switching and reduce system performance. This can increase the requirement to perform Ada optimization.

30.8.3 Object Oriented Development

The OOD methodology affects the "Requirements for Extensive Ada Optimization" problem in the following ways:

Ada-oriented: The OOD methodology directly encourages the use of Ada constructs such as packages, tasks, and generics. OOD provides good guidance in the use of these constructs to design Ada programs and leads to a more efficient use of these features than a non-Ada-oriented methodology. This can reduce the requirements for extensive Ada optimization.

Overuse of tasking: The OOD methodology directly uses the Ada task construct to address concurrent processing, interrupt handling, management of shared resources, and other functions. OOD strives to control system complexity by minimizing the number of tasks that are required to perform processing. This can reduce the requirements for extensive Ada optimization.

Design quality: The OOD methodology tends to produce designs which have loose data coupling (due to packaging) and which are communicationally cohesive (functions are related by references to same data). These factors can decrease the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This can reduce the requirements for extensive Ada optimization.

30.8.4 Structured Analysis and Design Technique

The SADT methodology affects the "Requirements for Extensive Ada Optimization" problem in the following ways:

Design quality: The SADT methodology tends to produce designs which have loose data coupling and which are sequentially cohesive (elements are partitions of system data flow diagram). These factors can decrease the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This can reduce the requirements for extensive Ada optimization.

30.8.5 PAMELA

The PAMELA methodology affects the "Requirements for Extensive Ada Optimization" problem in the following ways:

Ada-oriented: The PAMELA methodology directly encourages the use of Ada constructs such as packages, tasks, and generics. PAMELA provides good guidance in the use of these constructs to design Ada programs and leads to a more efficient use of these features than a non-Ada-oriented methodology. This can reduce the requirements for extensive Ada optimization.

Overuse of tasking: The PAMELA methodology directly uses the Ada task construct to address concurrent processing, interrupt handling, management of shared resources, and other functions. PAMELA heavily emphasizes the use of concurrency to perform its processing; thus, the design uses a large number of tasks. This can greatly increase the requirements to perform extensive Ada optimization.

Design quality: The PAMELA methodology tends to produce designs which have loose data coupling (due to packaging) and which are sequentially cohesive (elements are partitions of system data flow diagram). These factors can decrease the number and complexity of Ada context switches (with their associated inefficiencies) that are performed. This can reduce the requirements for extensive Ada optimization.

Process state: The PAMELA methodology provides the developer with information concerning process states. This information is useful in evaluating the impact of the proposed tasking approach to minimize the overhead. This can reduce the requirements for extensive Ada optimization.

Auto coding: The PAMELA methodology allows the developer to automatically generate Ada source code based on the program design specifications. As a rule, code that is automatically generated is less efficient than that of a hand-coded program. This could increase the requirement to perform extensive Ada optimization.

30.8.6 Warnier-Orr

The Warnier-Orr methodology affects the "Requirements for Extensive Ada Optimization" problem in the following ways:

Design quality: The Warnier-Orr methodology tends to produce an inefficient Ada design due to low program cohesion and lack of suitability for real-time systems development, which can cause increased Ada context switching and reduce system performance. This can increase the requirement to perform Ada optimization.

30.8.7 Jackson

The Jackson methodology affects the "Requirements for Extensive

Ada Optimization" problem in the following ways:

Design quality: The Jackson methodology tends to produce an inefficient Ada design due to low program cohesion and lack of suitability for real-time systems development, which can cause increased Ada context switching and reduce system performance. This can increase the requirement to perform Ada optimization.

30.14 DIFFICULTY IN PERFORMING SECURE PROCESSING FOR ADA SYSTEMS

Due to the current unavailability of a secure Ada operating system and the excessive overhead associated with the use of a secure Ada kernel to restrict system memory accesses, it is currently difficult to build a secure processing application in Ada.

The Ada language allows the applications programmer to perform run-time, and system level operations which increase the difficulty involved in protecting classified programs and data within the system. These operations include the creation, access, and destruction of objects at run-time, the use of address specifications to access particular memory locations; and the writing and execution of in-line assembly language programs.

Also, the RSL code is not only resident in the target environment, but runs in conjunction with the applications programs. Thus, to fully verify the security of an Ada system, the ,RSL code must be evaluated and certified as part of the system certification effort. This is difficult because most Ada applications programmers have little knowledge concerning the operation of the RSL.

The following methodology features affect this generic Ada problem (Problem #14):

- Information hiding
- Ada oriented
- Data visibility
- Design quality
- Program/Data structure

30.14.1 Yourdon

The Yourdon methodology affects the "Inability to Perform Secure Ada Processing" problem in the following ways:

Information hiding: The Yourdon methodology encourages the localization and logical/physical grouping of data, thus producing a design which reduces the scope of visibility for program data. This increases the ability to perform secure Ada processing.

Data visibility: The Yourdon methodology provides data visibility, which allows the developer to determine the accessibility of program data. This increases the ability to perform secure Ada processing.

Design quality: The Yourdon methodology provides strong guidelines for determining the quality of the design in the areas of modularity and localization and tends to produce a design which has loose data coupling. This increases the ability to perform secure Ada processing.

30.14.2 Functional Decomposition

The Functional Decomposition methodology affects the "Inability to Perform Secure Ada Processing" problem in the following ways:

Information hiding: The Functional Decomposition methodology encourages the localization of data, thus producing a design which reduces the scope of visibility for program data. This increases the ability to perform secure Ada processing.

Data visibility: The Functional Decomposition methodology provides data visibility, which allows the developer to determine the accessibility of program data. This increases the ability to perform secure Ada processing.

Design quality: The Functional Decomposition methodology tends to produce a design which is somewhat tightly data coupled. This decreases the ability to perform secure Ada processing.

30.14.3 Object Oriented Development

The OOD methodology affects the "Inability to Perform Secure Ada Processing" problem in the following ways:

Information hiding: The OOD methodology encourages the localization and logical/physical grouping of data through the use of Ada packages, thus producing a design which restricts data access. This increases the ability to perform secure Ada processing.

Ada-oriented: The OOD methodology leads directly to the use of Ada packages to encapsulate objects and the functions that operate on them, which leads to programs that are loosely data coupled. The loose data coupling restricts data access. This increases the ability to perform secure Ada processing.

Data visibility: The OOD methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This increases the ability to perform secure Ada processing.

Design quality: The OOD methodology tends to produce a design which has loose data coupling, due in large part to the concept of packaging objects and their related functions together. The loose data coupling restricts data access. This increases the ability to perform secure Ada processing.

Program/data structure: The OOD methodology tends to result in programs that are communicationally cohesive (elements related by reference to same data) and packaged data structures which reduce the amount of data shared between programs. The loose data coupling restricts data access. This increases the ability to perform secure Ada processing.

30.14.4 Structured Analysis and Design Technique

The SADT methodology affects the "Inability to Perform Secure Ada Processing" problem in the following ways:

Information hiding: The SADT methodology encourages the localization and logical/physical grouping of data through the use of Ada packages, thus producing a design which restricts data access. This increases the ability to perform secure Ada processing.

Data visibility: The SADT methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This increases the ability to perform secure Ada processing.

Design quality: The SADT methodology can produce a design which has loose data coupling. The loose data coupling restricts data access. This increases the ability to perform secure Ada processing.

30.14.5 PAMELA

The PAMELA methodology affects the "Inability to Perform Secure Ada Processing" problem in the following ways:

Information hiding: The PAMELA methodology encourages the localization and logical/physical grouping of data through the use of Ada packages, thus producing a design which restricts data access. This increases the ability to perform secure Ada processing.

Ada-oriented: The PAMELA methodology leads directly to the use of Ada packages to encapsulate objects and the functions that operate on them, which leads to programs that are loosely data coupled. The loose data coupling restricts data access. This increases the ability to perform secure Ada processing.

Data visibility: The PAMELA methodology provides data visibility, which allows the developer to assess and reduce the data coupling within the system. This increases the ability to perform secure Ada processing.

Design quality: The PAMELA methodology tends to produce a design which has loose data coupling, due in large part to the concept of packaging objects and their related functions together. The loose data coupling restricts data access. This increases the ability to perform secure Ada processing.

Program/data structure: The PAMELA methodology tends to result in programs that are sequentially cohesive (elements are partitions of system data flow) and packaged data structures which reduce the amount of data shared between programs. The loose data coupling restricts data access. This increases the ability to perform secure Ada processing.

30.14.6 Warnier-Orr

The Warnier-Orr methodology affects the "Inability to Perform Secure Ada Processing" problem in the following ways:

Information hiding: The Warnier-Orr methodology encourages the localization and logical/physical grouping of data, thus producing a design which reduces the scope of visibility for program data. This increases the ability to perform secure Ada processing.

Data visibility: The Warnier-Orr methodology provides data visibility, which allows the developer to determine the accessibility of program data. This increases the ability to perform secure Ada processing.

Design quality: The Warnier-Orr methodology tends to produce a design which has loose data coupling. This increases the ability to perform secure Ada processing.

30.14.7 Jackson

The Jackson methodology affects the "Inability to Perform Secure Ada Processing" problem in the following ways:

Information hiding: The Jackson methodology encourages the localization and logical/physical grouping of data, thus producing a design which reduces the scope of visibility for program data. This increases the ability to perform secure Ada processing.

Data visibility: The Jackson methodology provides data visibility, which allows the developer to determine the accessibility of program data. This increases the ability to perform secure Ada processing.

Design quality: The Jackson methodology tends to produce a design which has loose data coupling. This increases the ability to perform secure Ada processing.

30.15 DIVERSITY IN IMPLEMENTATION OF APSE's

The task of developing Ada software for computers integral to weapon systems is a complex one, and would be impossible without good support tools. The set of these tools to be used with an Ada compiler is called an Ada Program Support Environment (APSE), and the APSE has been the subject of much study since the Ada language was specified.

One of the problems recognized very early by both the Army Communications and Electronics Command (CECOM) and by the Air Force was the need for standardized and portable APSEs. The Air Force effort was lost in a funding problem soon after its inception, and the CECOM effort, which resulted in the Ada Language System (ALS), was terminated and made public domain. Sonicraft, which contracted Softech to retarget the ALS to the Intel 8086, tried the ALS and found tool performance below the range of usability.

This left the situation where each compiler vendor has marketed its own version of an APSE, with each requiring training both for users and for the host computer support engineers. This, in turn, has made it far more difficult to transition from one compiler to another during a project, a necessity all too often brought about by other Ada problems [Problem #17 for example].

The extent of this problem depends upon the complexity of the APSE that you now have and the one you are considering acquiring. If one of the APSEs is the ALS, the problem is very severe. Sonicraft had to send three VAX system support engineers to a two week course just to learn to install and configure the tools in the ALS. Users also needed training, which was given by these three engineers. Then, because of the extreme slowness of ALS operations (about a tenth as fast as comparable operations under the DEC VMS operating system), a major effort was required to try to "tweak" the ALS and the underlying VMS parameters to effect a speedup. This big investment in time and money was sacrificed when Sonicraft abandoned the ALS.

Col. Wm. Whitaker (Ret), commenting on a WIS report at the Washington Ada Symposium, stated that most programmers make do with a very minimal support environment, and that many of the exotic tools described in the literature either do not work or are not widely used (or both). One study [Hans85] defines the minimal tool set needed as a screen editor and an interactive debugger. You are indeed fortunate if your APSE contains a good source-level interactive debugger [Problem #9], but many APSEs contain tool sets which are quite complex, requiring training for all project designers and host support people.

The following methodology features affect this generic Ada problem (Problem #15):

- o Methodology tools

o Design quality

30.15.1 Yourdon

The Yourdon methodology affects the "Diversity In Implementation of APSEs" problem in the following ways:

Methodology tools: The Yourdon methodology tools can be integrated into a number of current APSEs. This reduces the impact of diversity in implementation of APSEs.

30.15.2 Functional Decomposition

The Functional Decomposition methodology affects the "Diversity In Implementation of APSEs" problem in the following ways:

Methodology tools: The Functional Decomposition methodology tools can be integrated into a number of current APSEs. This reduces the impact of diversity in implementation of APSEs.

30.15.3 Object Oriented Development

The OOD methodology affects the "Diversity In Implementation of APSEs" problem in the following ways:

Methodology tools: The OOD methodology tools can be integrated into a number of current APSEs. This reduces the impact of diversity in implementation of APSEs.

30.15.4 Structured Analysis and Design Technique

The SADT methodology affects the "Diversity In Implementation of APSEs" problem in the following ways:

Methodology tools: The SADT methodology tools can be integrated into a number of current APSEs. This reduces the impact of diversity in implementation of APSEs.

30.15.5 PAMELA

The PAMELA methodology affects the "Diversity In Implementation of APSEs" problem in the following ways:

Methodology tools: The PAMELA methodology tools can be integrated into a number of current APSEs. This reduces the impact of diversity in implementation of APSEs.

30.15.6 Warnier-Orr

The Warnier-Orr methodology affects the "Diversity In Implementation of APSEs" problem in the following ways:

Methodology tools: The Warnier-Orr methodology tools can be integrated into a number of current APSEs. This reduces the impact of diversity in implementation of APSEs.

30.15.7 Jackson

The Jackson methodology affects the "Diversity In Implementation of APSEs" problem in the following ways:

Design quality: The Jackson methodology tends to produce a design which has good localization of data. However, the program structure is created by mirroring the data structure, which can result in low cohesion and requires more internal memory management. This increases the impact of Ada memory management overhead.

30.17 DIFFERENCE IN BENCHMARKING ADA SYSTEMS

Benchmarks can be of two main types, those that are used to time and size portions of application code (usually using breadboard hardware) and those that are used to evaluate compilers and other support tools. It is the benchmark software for support tools that is addressed in this problem.

Ada is a very powerful, but also very new, programming language for embedded, mission-critical software. Whenever an application is being planned that is significantly different from previous experience in the software organization performing the task, benchmarking is normally relied upon to scope out the job. Unfortunately, the very constructs that make Ada a valuable embedded software language are hard to find in widely available benchmarks.

The benchmark most often cited by compiler vendors seems to be the Dhrystone, which has none of the new Ada constructs (tasking, interrupt handling, direct addressing, etc.) which make it superior to languages like Pascal for embedded systems. In comparison to real application code, such as the Sonicraft MEECN system, the Dhrystone benchmark is too optimistic in both compilation speed (lines of code per minute) and in object code size (bytes per line of source code). Errors could be a problem if they are not discovered until application code begins to emerge somewhere around the end of the Detail Design Phase. The project is supposed to be ready to code very heavily at that point, yet will find itself with undersized computer resources, both for the host and the target, if the Dhrystone numbers had been believed.

One approach taken to deal with this problem was reported by M. Kamrad of Honeywell at the Jan 87 SIGAda Conference. He recommended postponing the decision of how many processors to put in the system and what software functions run in each until the coding has matured enough that its final size and timing requirements are known.

According to D. L. Doty [Doty78], this can be a long wait. He has observed size-estimate errors greater than 100 percent at the RFP stage, 75 percent up to the Preliminary Design Review, and 50 percent up to the Critical Design Review.

But all this leaves the compiler vendor in a quandry if he is trying to introduce a new compiler. Unless he can test it against a real application in Ada which is already running under another Ada compiler, it will be difficult to convince potential customers that this new compiler can really handle an embedded Ada application.

The following methodology features affect this generic Ada problem:

- o Ada-oriented
- o Overuse of tasking
- o Problem definition
- o Design quality
- o Design consistency

30.17.1 Yourdan

The Yourdan methodology impacts the "Difference in Benchmarking Ada Systems" problem in the following ways:

Problem definition: The accurate model of the real world which results from the Yourdan methodology helps prevent massive rework late in the design cycle. This rework could affect the computer resource requirements, compounding any errors caused by poor benchmarks.

Design quality: The feedback on design quality provided by the Yourdan methodology reduces rework late in the design lifecycle. This rework could require additional computer resources, compounding the benchmarking problems.

Design consistency: The kind of software designs produced by the Yourdan methodology are very predictable once an application in the same family (i.e., radar warning receivers) of systems is done. This makes the computer resource needs more predictable, easing the errors caused by bad benchmarks.

30.17.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Difference in Benchmarking Ada Systems" problem in the following ways:

Design quality: The feedback on design quality provided by the Functional Decomposition methodology reduces cost/schedule overruns due to rework late in the design lifecycle. This rework could require additional computer resources, which may already be in short supply due to benchmarking errors.

Design consistency: The design produced by this methodology can vary over an enormous range because there are no checks provided. This can lead to estimation errors for computer resources which can make the benchmarking-induced errors worse.

30.17.3 Object Oriented Development

The OOD methodology impacts the "Difference in Benchmarking Ada Systems" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This decreases the chances for unexpected development tasks which could increase computer resource requirements, helping to keep the reserves available to make up shortages caused by benchmarking errors.

Problem definition: The designs from OOD match the real world problems quite closely, therby reducing the risk of major cost/schedule impact due to a small change in the problem definition. This makes it less risky to specify the computer resource needs earlier in the design cycle and helps to overcome the problems due to benchmarking errors.

Design quality: The feedback on design quality provided by the Object Oriented Design methodology reduces rework late in the design lifecycle. This avoids complications involving late changes in the computer resource requirements and helps the benchmark-caused shortages.

Design consistency: The design produced by this methodology is fairly consistent because there are checks provided. This makes it easier to predict computer resource needs early in the design and helps overcome sizing and timing errors from the benchmarking.

30.17.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Difference in Benchmarking Ada Systems" problem in the following ways:

Problem definition: Accurate symbology and modelling reduce the potential for disruptive rework late in the design cycle. This reduces the risk of disruptive computer resource increases being needed late in the program, which would compound the benchmarking problems.

Design quality: The feedback on design quality provided by the Structured Analysis and Design Technique methodology reduces rework late in the design lifecycle. This, in turn, reduces risk of demands for more memory or cpu speed, helping the benchmarking problem.

Design consistency: The design produced by this methodology varies over a limited range because of the checks provided. This makes computer resource needs more predictable, which helps offset any errors caused by benchmarks.

30.17.5 PAMELA

The PAMELA methodology impacts the "Difference in Benchmarking Ada Systems" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This stabilizes the coding step of the development process and makes the computer resource needs less uncertain. This helps overcome benchmark-induced errors.

Over-use of tasking: PAMELA produces Ada designs which very heavily use the tasking construct. In some present Ada compilers it is necessary to "tune" the run-time support library default task stack size so that the total allocated task stacks do not consume all available RAM. This causes uncertainty in memory needs. However, tasking execution speed is improving with more recent compilers, which allows trades of speed for memory to be made in many cases. These two uncertainties tend to offset each other, but nevertheless add variability to the estimates of computer resource requirements that compounds the variability from benchmarking errors.

Design quality: The feedback on design quality provided by the PAMELA methodology reduces rework late in the design lifecycle. This reduces the risk of late changes to the computer resource requirements, alleviating the benchmarking problems.

Design consistency: PAMELA, being a highly-automated methodology, produces extremely consistent designs that make the specification of the computer resource requirements much safer.

30.17.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Difference in Benchmarking Ada Systems" problem in the following ways:

Design quality: The feedback on design quality provided by the Warnier-Orr methodology is not as strong as with other methodologies, so computer resource requirements changes late in the design lifecycle could occur more frequently. This adds to the benchmarking problems.

Design consistency: The design produced by this methodology is restricted to a narrow range because the process is so simple. This reduces computer resource specification problems that could impact cost and schedule and aggrevate the benchmark errors.

30.17.7 Jackson

The Jackson methodology impacts the "Difference in Benchmarking Ada Systems" problem in the following ways:

Problem definition: The extraordinary attention paid by Jackson to insure that the design models the real world as closely as possible helps to ensure that a small specification change in the real environment results in a correspondingly small change in the software design. This tends to stabilize computer resource needs, helping to overcome benchmark errors.

Design quality: The guarantee of design quality provided by the Jackson methodology reduces risks of computer resource changes due to rework late in the design lifecycle. This helps to offset any benchmark-induced errors.

Design consistency: Jackson is one of the most consistent of the common methodologies, making it easier to specify cpu speed and memory requirements early in the design cycle. This reduces the impact of computer resource estimation errors caused by the use of inappropriate benchmarks.

30.18 LACK OF ADA SOFTWARE DEVELOPMENT TOOLS

Although a number of software tools have been developed for use in Ada environments, there is still a variety of tools that are desirable to further improve the productivity and performance associated with Ada development efforts. Some of these tools are currently in various stages of development (planning, design, implementation, testing).

Some of the tools that would be useful for Ada efforts include:

* Ada Design Generators
* Ada Code Generators
* Ada Source Code Analyzers
* Ada-oriented Debuggers
* Ada Syntax-directed Editors
* Ada Cost Estimation Models
* Ada Project Management Tools
* Secure Ada Operating Systems
* Automated Ada Test Tools

The lack of Ada tools affects the overall productivity of development efforts. The availability of these tools would decreases the number of development activities that are currently performed manually. It would also reduce the overall development effort by reducing the requirement for applications programmers to develop their own project-specific tools. The availability would also improve the consistency of the products developed on Ada projects and would help impose and enforce project design methods and development standards.

The following methodology features affect this generic Ada problem:

- o Ada-oriented
- o Methodology tools

30.18.1 Yourdan

The Yourdan methodology impacts the "Lack of Ada Software Development Tools" problem in the following way:

Methodology tools: Yourdan tools are readily available from more than one vendor. The tools are mature, having been released over ten years ago and modified several times since then. This improves the productivity during the development process.

30.18.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Lack of Ada Software Development Tools" problem in the following way:

Methodology tools: Functional Decomposition tools are general-purpose packages readily available from more than one vendor. The tools are mature. This makes the development process more efficient.

30.18.3 Object Oriented Development

The OOD methodology impacts the "Lack of Ada Software Development Tools" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This makes the coding step of the development process much more efficient.

Methodology tools: OOD tools are available from just one vendor. The tools are still not mature. This single source helps make the development process more efficient, but the immaturity makes it more risky.

30.18.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Lack of Ada Software Development Tools" problem in the following way:

Methodology Tools: SADT users can adopt many of the same general-purpose diagramming tools that Yourdan and other methodologies have caused to be developed. The widespread availability of these tools increases productivity during the development cycle.

30.18.5 PAMELA

The PAMELA methodology impacts the Lack of Ada Software Development Tools" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This simplifies the coding step of the development process, increasing the productivity and requiring less general-purpose tool support.

Methodology tools: PAMELA tools are available from just one vendor. The tools are still not mature. This lack of competition does not affect the productivity improvements, but the immaturity could offset this by driving changes in future (improved) releases.

30.18.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Lack of Ada Software Development Tools" problem in the following way:

Methodology Tools: Warnier-Orr tools mainly help with the graphics used in the design documentation, which is only one of many Ada support tools needed. However, this activity consumes a large percentage of a developer's hours, so it is important.

30.18.7 Jackson

The Jackson methodology impacts the "Lack of Ada Software Development Tools" problem in the following way:

Methodology tools: Jackson needs are satisfied by general-purpose tools which are mature, having been released over ten years ago and modified several times since then. This reduces the cost and schedule estimations for projects using these tools.

30.19 ADA LANGUAGE COMPLEXITY

The complexity of the Ada language makes it difficult to learn, use effectively, and test (validate). The Ada language requires extensive training for programmers to learn language syntax, proposed development methods, software engineering standards, and implementation issues for real-time embedded systems.

The Ada language is also difficult to use effectively and efficiently. For example, the current inefficiency of Ada compilers creates an environment where unchecked use of certain Ada features (tasking, generics, etc..) can cause significant impacts on system performance. Also, Ada development methods and programming standards/conventions are still being established.

The power and complexity of the Ada programming language can also make it difficult to test and validate an Ada system. The functional and performance impacts of the RSL, a very complex piece of software, play an important part in the testing process.

The following methodology features affect this generic Ada problem:

- o Ada-oriented
- o Ease of use
- o Auto coding

30.19.1 Yourdan

The Yourdan methodology impacts the "Ada Language Complexity" problem in the following way:

Ease of use: Good tools make Yourdan even easier to use, but the basic methodology is not very complicated. This allows new project personnel to operate reasonably effectively even before completing a training course in this methodology. This allows the focus to be on Ada complexities.

30.19.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Ada Language Complexity" problem in the following way:

Ease of use: Using tools make Functional Decomposition even simpler, but the basic methodology is not very complicated. This allows newly-assigned programmers to be more productive during their first few months, concentrating on Ada-related issues.

30.19.3 Object Oriented Development

The OOD methodology impacts the "Ada Language Complexity" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This speeds the coding step of the development process and directly reduces the amount of Ada training required. Also, because of the close tie-in to Ada, learning the OOD methodology can make the task of learning Ada much easier.

Ease of use: The tools make OOD easier to use, but the basic methodology is not very complicated. This cuts the level of confusion of new programmers since Ada is already a challenge facing them.

30.19.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Ada Language Complexity" problem in the following way:

Ease of use: The methodology is fairly complex, delaying newly-assigned programmers from becoming productive. If they don't know Ada, it will be quite a while before they become productive team members.

30.19.5 PAMELA

The PAMELA methodology impacts the "Ada Language Complexity" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This eliminates the need to teach the coding step of the development process.

Ease of use: The tools make PAMELA easier to use, but the basic methodology is not very complicated. The combination of automating some functions and simplifying others makes PAMELA very attractive where Ada programming experience is a problem.

Auto coding: This would normally be an advantage because it simplifies a programmer's task, but PAMELA generates constructs that are inefficient in some current Ada compilers. This could force redesign late in the development cycle, making the Ada complexity problems actually get worse.

30.19.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Ada Complexity Problem" problem in the following way:

Ease of use: Warnier-Orr is a simple methodology, but the symbology is unique and will take some time to become assimilated. This compounds the problem of Ada complexity during the first few months.

30.19.7 Jackson

The Jackson methodology impacts the "Ada Language Complexity" problem in the following way:

Ease of use: The tools make Jackson easier to use, but the basic methodology is quite complicated. This extends the period of time before newly-assigned engineers become productive, compounding the Ada complexity problem.

30.20 CUSTOMIZATION OF RUN-TIME SUPPORT LIBRARY

When an Ada compiler and its respective run-time support code is validated, it is on a target system specified by the compiler developer. The problem is that an applications user of the compiler will usually have a different configuration (memory map, I/O map) for their particular system. Also, the user may desire different default values for their run-time application, such as the task stacks. Instead of a four kilobyte default, as experienced on Sonicraft's MEECN project they may elect to have a three kilobyte default. Therefore, the applications user must modify or customize the run-time support code. To get this type of customization the user will need to recompile the run-time support library. If the user desires to perform this task himself, he must usually purchase the source code rights for the RSL and train some people on how to perform the required task. The other option is to contract the vendor to make the required modifications. Either course of action will add more cost to the development of the application program.

Sonicraft discovered on its own Ada projects that the ALS run-time software needed to be reconfigured for memory map allocation, interrupt vector addresses, and Input/Output addresses for Intel's PIC and PIT chips. Further, RSL code had to be repackaged to allow for better selective linking and modified to remove unused 8087 code and to reduce interrupt overhead.

The following methodology features affect this generic Ada problem:

- o Ada-oriented
- o Overuse of tasking
- o Problem definition
- o Design quality
- o Design consistency

30.20.1 Yourdan

The Yourdan methodology impacts the "Customization of Run-Time Support Library" problem in the following ways:

Problem definition: The accurate model of the real world which results from the Yourdan methodology helps prevent massive rework late in the design cycle. This rework could affect the requirements for the run-time support library when there is really no time to fix it.

Design quality: The feedback on design quality provided by the Yourdan methodology reduces rework late in the design lifecycle. This rework could force run-time support library changes serious enough to jeopardize the project.

Design consistency: The kind of software designs produced by the Yourdan methodology are very predictable once an application in the same family (i.e., radar warning receivers) of systems is done. This makes the early design of the run-time support system an acceptable risk.

30.20.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Customization of Run-Time Support Library" problem in the following ways:

Design quality: The feedback on design quality provided by the Functional Decomposition methodology reduces cost/schedule overruns due to rework late in the design lifecycle. This rework, if extensive enough, can affect the run-time support library design.

Design consistency: The design produced by this methodology can vary over an enormous range because there are no checks provided. This makes run-time support library requirements hard to predict early in the program lifecycle when it is best to commit them.

30.20.3 Object Oriented Development

The OOD methodology impacts the "Customization of Run-Time Support Library" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This decreases the chances for unexpected development tasks which could drive run-time support library changes.

Problem definition: The designs from OOD match the real world problems quite closely, therby reducing the risk of major cost/schedule impact due to a small change in the problem definition. This makes it less risky to design the run-time support library early in the project.

Design quality: The feedback on design quality provided by the OOD methodology reduces rework late in the design lifecycle. This avoids complications involving late changes in the run-time support library.

Design consistency: The design produced by this methodology is fairly consistent because there are checks provided. This makes run-time support library requirements more predictable.

30.20.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Customization of Run-Time Support Library" problem in the following ways:

Problem definition: Accurate symbology and modelling reduce the potential for disruptive rework late in the design cycle. This reduces the risk of disruptive run-time support library changes being needed late in the program.

Design quality: The feedback on design quality provided by the Structured Analysis and Design Technique methodology reduces rework late in the design lifecycle. This, in turn, reduces risk of run-time support library redesign late in the design.

Design consistency: The design produced by this methodology varies over a limited range because of the checks provided. This makes run-time support library needs more predictable.

30.20.5 PAMELA

The PAMELA methodology impacts the "Customization of Run-Time Support Library" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This stabilizes the coding step of the development process and makes the run-time support library requirements more predictable.

Over-use of tasking: PAMELA produces Ada designs which very heavily use the tasking construct. In some present Ada compilers it is necessary to "tune" the run-time support library default task stack size so that the total allocated task stacks do not consume all available RAM.

Design quality: The feedback on design quality provided by the PAMELA methodology reduces rework late in the design lifecycle. This reduces the risk of late changes to the run-time support library.

Design consistency: PAMELA, being a highly-automated methodology, produces extremely consistent designs that make the specification of the run-time support library much safer.

30.20.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Customization of Run-Time Support Library" problem in the following ways:

Design quality: The feedback on design quality provided by the Warnier-Orr methodology is not as strong as with other methodologies, so run-time support library rework late in the design lifecycle could occur more frequently.

Design consistency: The design produced by this methodology is restristed to a narrow range because the process is so simple. This reduces run-time support library specification problems that could impact cost and schedule.

30.20.7 Jackson

The Jackson methodology impacts the "Customization of Run-Time Support Library" problem in the following ways:

Problem definition: The extraordinary attention paid by Jackson to insure that the design models the real world as closely as possible helps to ensure that a small specification change in the real environment results in a correspondingly small change in the software design. This tends to stabilize run-time support library specifications.

Design quality: The guarantee of design quality provided by the Jackson methodology reduces risks of run-time support libary rework late in the design lifecycle.

Design consistency: Jackson is one of the most consistent of the common methodologies, making it easier to specify run-time support library requirements.

30.21 LACK OF EXPERIENCED ADA PROGRAMMERS

As we in the Ada community have heard so often, Ada is "not just another high order language", but is a whole new approach to software design. Indeed, many respected individuals feel that the major contribution that Ada will make is to train programmers in the use of modern software design techniques [Broo87]. When we speak of the problem of not enough experienced Ada programmers, it is in this broader context that we see the problem.

Experience at Sonicraft is that a fresh graduate with a CS degree can learn Ada syntax well enough in about three weeks to start making useful contributions to a project. Despite the complaints about Ada being "too complex", it turns out in practice that some of the complex features need only be used by a small percentage of design team members.

Teaching a design method takes far longer in the experience of Sonicraft. Formal courses typically go for two or three weeks at two or three hours per day (with homework), but it takes actual project experience before most people really understand the value of a design method and become advocates of it. Without this kind of full support, most design methods are reduced to being an extra paperwork burden in the minds of the programmers.

Formal courses are another problem because hiring is usually done over a period of time, rather than hiring the first few applicants who meet the minimum qualifications. Putting untrained people on the job while they are waiting for the next class to start brings on the infamous Brooks' Law effects, where the addition of additional programmers slows down the team [Broo75]. This happens because the experienced people must spend considerable time explaining to the new people some of the things they would normally have learned in the classes.

Finally, the supply of experienced Ada programmers is not easy to measure. At the November 86 SIGAda Conference, for example, it was reported that the biggest shortage in Ada-trained people is among managers. The supply of programmers was greater than the number of people actually doing Ada work. This result flies in the face of experience in hiring at Sonicraft, where over a four year period over forty designers were hired to do Ada work and only one of them had any Ada experience on the job. Very few of the CS majors had a course that even used Ada until about 1986.

This apparent discrepancy between the reported oversupply and the experienced shortage of Ada programmers could have been a local shortage or it could have been pure chance. But it also could have been caused by large firms training massive numbers of people in Ada in anticipation of future Ada contracts. This would indeed cause an oversupply to be measured if one counted everyone who went through these courses as an experienced Ada programmer.

The following methodology features affect this generic Ada problem:

- o Ada-oriented
- o Ease of use
- o Ease of learning

30.21.1 Yourdan

The Yourdan methodology impacts the "Lack of Experienced Ada Programmers" problem in the following ways:

Ease of use: The good tools make Yourdan even easier to use, but the basic methodology is not very complicated. This allows new project personnel to operate reasonably effectively even before completing a training course.

Ease of learning: Training courses have been conducted by Yourdan for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This cuts training costs, allowing more effort to be devoted to Ada.

30.21.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Lack of Experienced Ada Programmers" problem in the following ways:

Ease of use: Using tools make Functional Decomposition even simpler, but the basic methodology is not very complicated. This allows newly-assigned programmers to be more productive during their first few months.

Ease of learning: Training courses have been conducted by several vendors for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This cuts training costs and allows training efforts to focus more on Ada.

30.21.3 Object Oriented Development

The OOD methodology impacts the "Lack of Experienced Ada Programmers" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This speeds the coding step of the development process and directly reduces the amount of Ada training required. Also, because of the close tie-in to Ada, learning the OOD methodology can make the task of learning Ada much easier.

Ease of use: The tools make OOD easier to use, but the basic methodology is not very complicated. This cuts the level of confusion of new programmers since Ada is already a challenge facing them.

Ease of learning: Training courses are offered frequently. A good book and a sequel have also been written on this methodology that assist in understanding how it works. This cuts training costs for the OOD methodology.

30.21.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Lack of Experienced Ada Programmers" problem in the following ways:

Ease of use: The methodology is fairly complex, delaying newly-assigned programmers from becoming productive. If they don't know Ada, it will be quite a while before they become productive team members.

Ease of learning: SADT is probably best taught using in-house courses because SADT is very well documented but course offerings are infrequent. This increases the burden on the cadre for the project since Ada will also draw upon their training resources.

30.21.5 PAMELA

The PAMELA methodology impacts the "Lack of Experienced Ada Programmers" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This eliminates the need to teach the coding step of the development process.

Ease of use: The tools make PAMELA easier to use, but the basic methodology is not very complicated. The combination of automating some functions and simplifying others makes PAMELA very attractive where Ada programming experience is a problem.

Ease of learning: Training courses are offered frequently. A good book has also been written on this methodology that assists in understanding how it works. This cuts the training burden on the staff.

30.21.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Lack of Experienced Ada Programmers" problem in the following ways:

Ease of use: Warnier-Orr is a simple methodology, but the symbology is unique and will take some time to become assimilated. This compounds the problem of low productivity during the first few months because Ada is unfamiliar.

Ease of learning: Warnier-Orr is an established methodology that has been in use over ten years, and there is a wealth of documentation available to help learn to use it. Training course availability is no longer adequate, however, so training burden on the staff could be a serious problem.

30.21.7 Jackson

The Jackson methodology impacts the "Lack of Experienced Ada Programmers" problem in the following ways:

Ease of use: The tools make Jackson easier to use, but the basic methodology is quite complicated. This extends the period of time before newly-assigned engineers become productive, compounding the Ada problem.

Ease of learning: Training courses have been conducted for over ten years and are still offered. Several good books have also been written on this methodology that assist in understanding how it works. This cuts training costs and permits staff attention to be focussed on Ada.

30.22 EXTENSIVE ADA TRAINING REQUIREMENTS

The complexity of the Ada programming language and the system development and run-time environments results in extensive training requirements for Ada applications programmers.

To fully prepare the applications to develop Ada software a variety of training should be provided at different levels throughout the various phases of the project. Training should be provided at a minimum at the following levels - Senior Technical Staff, Junior Technical Staff, and Management.

The training courses to be offered should be selected according to the type and level of personnel being trained. The technical staff members should be offered all or some of the following courses:

* Ada Language Overview
* Advanced Ada Language Issues
* Ada Development Methods
* Ada Implementation Issues
* Ada APSE Issues

The management staff should be offered the following courses:

* Ada Language Overview
* Ada Cost/Schedule Estimation and Tracking
* Ada Development Environment
* Ada Productivity Issues

Ada training costs are high and a large amount of time is required to train the programmers. From 3 - 12 weeks should be allotted for the technical staff and from 1-3 weeks for the management staff. The training courses should be scheduled to coincide with the proposed project staffing plan. It should be noted that it is more difficult to retrain non-Ada programmers than to train new programmers.

The following methodology features affect this generic Ada problem:

o Ease of use
o Ease of learning

30.22.1 Yourdan

The Yourdan methodology impacts the "Extensive Ada Training Requirements" problem in the following ways:

Ease of use: Good tools make Yourdan even easier to use, but the basic methodology is not very complicated. This allows attention to be focussed on Ada-related issues.

Ease of learning: Training courses have been conducted by Yourdan for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This reduces the training effort required by the prospective user.

30.22.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Extensive Ada Training Requirements" problem in the following ways:

Ease of use: Using tools make Functional Decomposition even simpler, but the basic methodology is not very complicated. This provides a stable framework in which to learn Ada.

Ease of learning: Training courses have been conducted by several vendors for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This cuts the training effort required by project personnel.

30.22.3 Object Oriented Development

The OOD methodology impacts the "Extensive Ada Training Requirements" problem in the following ways:

Ease of use: The tools make OOD easier to use, but the basic methodology is not very complicated. This reduces the time it takes new personnel to become productive.

Ease of learning: Training courses are offered frequently. A good book and a sequel have also been written on this methodology that assist in understanding how it works. This cuts training effort by the project cadre.

30.22.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Extensive Ada Training Requirements" problem in the following ways:

Ease of use: The methodology is fairly complex, delaying the new personnel from giving needed attention to learning Ada.

Ease of learning: The SADT methodology is well documented in the literature and some training courses are offered. In-house expertise would be needed to assure training is available when needed for new personnel, compounding the Ada training problem.

30.22.5 PAMELA

The PAMELA methodology impacts the "Extensive Ada Training Requirements" problem in the following ways:

Ease of use: The tools make PAMELA easier to use, but the basic methodology is not very complicated. This helps put the focus on Ada design issues.

Ease of learning: Training courses are offered frequently. A good book has also been written on this methodology that assists in understanding how it works. This cuts training effort by project leaders.

30.22.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Extensive Ada Training Requirements" problem in the following ways:

Ease of use: Warnier-Orr is a simple methodology, but the symbols and graphics will take some time to feel natural. This diverts attention from Ada-related design issues.

Ease of learning: Warnier-Orr is an established methodology that has been in use over ten years, and there is a wealth of documentation available to help learn to use it. Training course availability is no longer adequate, however, so training effort by project personnel would be high.

30.22.7 Jackson

The Jackson methodology impacts the "Extensive Ada Training Requirements" problem in the following ways:

Ease of use: The tools make Jackson easier to use, but the basic methodology is quite complicated. This diverts the attention from Ada issues, complicating the problem very seriously.

Ease of learning: Training courses have been conducted for over ten years and are still offered. Several good books have also been written on this methodology that assist in understanding how it works. The complexity of the underlying material, however, could burden project training efforts.

30.23 INACCURACY OF C/S ESTIMATE FOR ADA PROGRAM

Most of the Ada problems recounted here cause cost/schedule perturbations for embedded mission-critical applications [Problems #1,3,4,5,6,7,8,9,10,11,13,14,15,16,17,18,19,20,21,22, 23,24,25,26].

Because of the pervasiveness of the influence of almost every problem on cost and schedule performance, the task of estimating cost and schedule performance becomes even more complicated. Not only must you understand enough about software cost/schedule estimating, but you must also understand enough about the additional problems you get in Ada. This Ada problem knowledge is a very rare commodity today, which is one of the reasons this report is being written.

In an effort to help, there have been extensions to one of the most popular cost models, COCOMO [Boeh81], that attempt to account for some of the better-known Ada problems [Jens85]. These include the instability of the compiler (major revisions every six months being the present norm) and the extra time it takes to train Ada programmers.

Even someone who has lived through the Ada problems might be hard-pressed to estimate their cost/schedule effects on a new project. The only real way to do this is to have someone in control of the project who understands the pitfalls well enough to avoid them, making their cost/schedule effects near zero (since avoidance costs are usually small).

But not all Ada-specific cost/schedule factors can be called problems. After all, the ultimate reason to use Ada is to save money, not to lose it. When a steady-state condition is reached after a few years of using Ada, developers should find the costs much improved. But again, since very few have reached this state, the actual gains are very difficult to estimate.

But even when all Ada-specific cost/schedule influences can be accounted for, which may be years in the future for many applications developers, the job of software cost/schedule estimating is anything but easy. This is a long-standing software engineering problem, with even estimation models with many years of good service being criticized for making errors of 100 percent or more [Keme87]. In fact, some recommend collecting your own statistics for several project done using your own programming support environment rather than using the factors in the models [Davi85]. Published model factors are based on data from hundreds of projects, but if your environment is significantly different than the industry norm then your responsiveness to these factors may indeed be unique.

The following methodology features affect this generic Ada problem:

- Ada-oriented
- Methodology tools
- Ease of use
- Ease of learning
- Problem definition
- Design quality
- Auto coding
- Design consistency

30.23.1 Yourdan

The Yourdan methodology impacts the "Inaccuracy of Cost/Schedule Estimates for Ada Programs" problem in the following ways:

Methodology tools: Yourdan tools are readily available from more than one vendor. The tools are mature, having been released over ten years ago and modified several times since then. This competition keeps acquisition costs low, and the maturity keeps development costs predictable.

Ease of use: The good tools make Yourdan even easier to use, but the basic methodology is not very complicated. This cuts rework costs and schedule slips due to misuse of the methodology.

Ease of learning: Training courses have been conducted by Yourdan for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This minimizes surprise training costs.

Problem definition: The accurate model of the real world which results from the Yourdan methodology helps prevent unexpected costs and schedule slips due to massive rework late in the design cycle.

Design quality: The feedback on design quality provided by the Yourdan methodology reduces costs and schedule impacts due to rework late in the design lifecycle.

30.23.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Inaccuracy of Cost/Schedule Estimates for Ada Programs" problem in the following ways:

Methodology tools: Functional Decomposition uses general-purpose tools readily available from more than one vendor. The tools are mature. This competition and the maturity keep development costs more stable.

Ease of use: Using tools make Functional Decomposition even simpler, but the basic methodology is not very complicated. This cuts unplanned rework costs and schedule slips due to misuse of the methodology.

Ease of learning: Training courses have been conducted by several vendors for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This cuts training cost surprises.

Design quality: The feedback on design quality provided by the Functional Decomposition methodology reduces cost/schedule overruns due to rework late in the design lifecycle.

Design consistency: The design produced by this methodology can vary over an enormous range because there are no checks provided. This increases the chances of miscommunications which could cause cost/schedule impact.

30.23.3 Object Oriented Development

The OOD methodology impacts the "Inaccuracy of Cost/Schedule Estimates for Ada Programs" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This speeds the coding step of the development process, decreasing the chances for unexpected development tasks.

Methodology tools: OOD tools are available from just one vendor. The tools are still not mature. This lack of competition destabilizes acquisition costs, and the immaturity keeps development costs at risk.

Ease of use: The tools make OOD easier to use, but the basic methodology is not very complicated. This cuts rework cost/schedule impacts due to misuse of the methodology.

Ease of learning: Training courses are offered frequently. A good book and a sequel have also been written on this methodology that assist in understanding how it works. This cuts training cost/schedule uncertainty.

Problem definition: The designs from OOD match the real world problems quite closely, therby reducing the risk of major cost/schedule impact due to a small change in the problem definition.

Design quality: The feedback on design quality provided by the OOD methodology reduces cost/schedule impact due to rework late in the design lifecycle.

Design consistency: The design produced by this methodology is fairly consistent because there are checks provided. This controls communication costs in the development group.

30.23.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Inaccuracy of Cost/Schedule Estimates for Ada Programs" problem in the following ways:

Ease of use: The methodology is fairly complex, increasing the risk of underbid development costs compared to alternative methodologies.

Problem definition: Accurate symbology and modelling reduce the potential for disruptive rework late in the design cycle.

Design quality: The feedback on design quality provided by the Structured Analysis and Design Technique methodology reduces cost/schedule increases due to rework late in the design lifecycle.

Design consistency: The design produced by this methodology varies over a limited range because of the checks provided. This controls communication costs.

30.23.5 PAMELA

The PAMELA methodology impacts the "Inaccuracies of Cost/Schedule Estimates for Ada Programs" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This stabilizes the coding step of the development process.

Methodology tools: PAMELA tools are available from just one vendor. The tools are still not mature. This lack of competition destabilizes acquisition costs, and the immaturity keeps development costs at risk.

Ease of use: The tools make PAMELA easier to use, but the basic methodology is not very complicated. This cuts rework risks due to misuse of the methodology.

Ease of learning: Training courses are offered frequently. A good book has also been written on this methodology that assists in understanding how it works. This cuts training cost risk.

Design quality: The feedback on design quality provided by the PAMELA methodology reduces cost/schedule impacts due to rework late in the design lifecycle.

Auto coding: PAMELA produces Ada code directly from the input design information, eliminating the often expensive step of trying to get a good compilation of Ada code written. If the Ada compiler happens to be slow, this can drastically improve Ada productivity.

Design consistency: The design produced by this methodology is fairly consistent because there are checks provided. This controls communication costs in the development group.

30.23.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Inaccuracy of Cost/Schedule Estimates for Ada Programs" problem in the following ways:

Ease of learning: Warnier-Orr is an established methodology that has been in use over ten years, and there is a wealth of documentation available to help learn to use it. Training course availability is no longer adequate, however, so training costs could be at risk until an in-house instructor becomes available.

Design quality: The feedback on design quality provided by the Warnier-Orr methodology is not as strong as with other methodologies, so cost/schedule problems due to rework late in the design lifecycle could occur more frequently.

Design consistency: The design produced by this methodology is restristed to a narrow range because the process is so simple. This reduces communication problems in the development team that could impact cost and schedule.

30.23.7 Jackson

The Jackson methodology impacts the "Inaccuracy of Cost/Schedule Estimates for Ada Programs" problem in the following ways:

Methodology tools: Jackson tools are general-purpose charting aids. The tools are mature, with many available ten years ago and modified several times since then. Changes that could drive cost/schedule increases are not likely.

Ease of use: The tools make Jackson easier to use, but the basic methodology is quite complicated. This increases rework cost/schedule risk due to misuse of the methodology.

Ease of learning: Training courses have been conducted for over ten years and are still offered. Several good books have also been written on this methodology that assist in understanding how it works. This cuts the uncertainty in training costs and schedules.

Design quality: The guarantee of design quality provided by the Jackson methodology reduces risks to cost and schedule due to rework late in the design lifecycle.

Design Consistency: Jackson is one of the most consistent of the common methodologies, cutting communication-induced cost and schedule problems for the team.

30.24 LACK OF ESTABLISHED ADA SOFTWARE DEVELOPMENT METHOD

One of the most important features of Ada is that it encourages the use of modern software engineering development practices [Problem #21]. These practices can best be exploited when they are packaged within a good development method. The lack of an established Ada development method can be a problem in that engineers trained in one method are less mobile if other organizations (even within the same parent organization) use a different one. This in turn contributes to the lack of experienced Ada programmers [Problem #21] since an important part of their experience is their design method training.

This is not to say that there is a lack of good Ada design methods. In fact there are two very good ones, Object Oriented Design (OOD) and PAMELA (TM George Cherry). One of these, OOD, is gaining both popularity and respect very rapidly, and PAMELA, while held back now by limitations described below, has good long-term potential also.

OOD is predicated upon the premise that problem definition is the first and hardest task a designer confronts. OOD concentrates on helping the designer with this task by stripping away much of the syntactic material that adds little to this task. It does this by defining the Object as an Ada package or task, which is a higher level view than the module of Yourdan [Boyd86]. This has led some to procla'm OOD as being the most promising of the "technical fads" now available to improve programmer productivity [Broo87]. But it also causes some to complain that OOD is too limited when it comes to expressing the dynamism of Ada systems in operation [Boyd86].

PAMELA, on the other hand, is best at describing the dynamism of Ada systems. It is the first process-flow design method specifically adapted to the Ada syntax [Boyd86], using easy to learn graphical techniques to input the design information that it will use to generate code automatically. This labor-saving step is also its biggest problem right now, however, since it generates an Ada task for every "single-thread" (no children) process [Boyd86]. With current Ada compilers the overhead due to context switching is too high to allow free use of tasking.

Sonicraft, in the MEECN system development in Ada, was forced to redesign extensively to cut down the number of tasks in the system because it could not meet the design constraints. Even the delivered design, with only ten tasks, was a problem because some were nested as much as three levels deep, causing very high overhead with the compiler then in use. This is the other objection to PAMELA: it leads to deeply-nested structures [Boyd86].

Note that neither of the two limitations to the use of PAMELA are necessarily long-lasting ones. When Ada compilers become available that can do very rapid context switches (perhaps in conjunction with underlying microprocessors that have richer instruction sets) and that maintain a single task stack for the parent task and all direct descendants (to avoid context switches among them), PAMELA may become much more widely used.

As a final note, there are some software organizations that service customers outside the Department of Defense and thus may need to use languages other than Ada on some projects. Sonicraft, for example, is currently involved in the FAA's Airport Modernization Program, and was specifically directed to use the same language as the Principal Contractor, which happened to be "C." Since one of the main goals of a functional software organization is the balancing of labor resources between on-going and planned projects, the use of a non-language-specific design method was preferred. This allowed programmers to be moved between the two major projects to satisfy special project needs or to further individual career goals. Had Sonicraft faced major retraining costs, because the design method was changed as well as the language, such moves would have been far less frequent.

The following methodology features affect this generic Ada problem:

- o Ada-oriented
- o Ease of use
- o Ease of learning
- o Traceability
- o Auto coding

30.24.1 Yourdan

The Yourdan methodology impacts the "Lack of Established Ada Software Development Methodology" problem in the following ways:

Ease of use: Good tools make Yourdan even easier to use, but the basic methodology is not very complicated. This makes organizations which have never used a common methodology to be more willing to give Yourdan a try.

Ease of learning: Training courses have been conducted by Yourdan for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This cuts training costs, one of the biggest barriers to the adoption of a standard methodology.

Traceability: The careful step-by-step approach outlined in Yourdan make it easy to see where each module is in the development cycle. This is an important feature for a methodology hoping to become a standard for the Ada community.

30.24.2 Functional Decomposition

The Functional Decomposition methodology impacts the

"Lack of Established Ada Software Development Methodology" problem in the following ways:

Ease of use: The basic Functional Decomposition methodology is not very complicated and follows an intuitively obvious approach. This makes organizations more comfortable in adopting it, especially those organizations with little prior experience in using a methodology.

Ease of learning: Training courses have been conducted by several vendors for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This avoids training problems and makes Functional Decomposition more attractive.

Traceability: The discreet milestones typical of this methodology make it easy to trace design progress, making this methodology mor popular among prospective buyers who are managers.

30.24.3 Object Oriented Development

The OOD methodology impacts the "Lack of Established Ada Software Development Methodology" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This simplifies the coding step of the development process, making OOD a stronger candidate for adoption as a standard Ada development methodology.

Ease of use: Tools make OOD easier to use, but the basic methodology is not very complicated. This helps organizations make the decision to adopt OOD.

Ease of learning: Training courses are offered frequently. A good book and a sequel have also been written on this methodology that assist in understanding how it works. The favorable publicity from the books also generates enthusiam among potential using organizations.

Traceability: OOD provides a less formal approach than some of the other methodologies, being structured mainly for the very early design phases. The resultant problems in tracing project progress through completion argue against the adoption of OOD as the standard Ada methodology.

30.24.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Lack of Established Ada Software Development Methodology" problem in the following ways:

Ease of use: The methodology is fairly complex, making it less attractive compared to alternative methodologies.

Ease of learning: The complexity of the methodology also makes it harder to learn, making it less likely to be adopted as the Ada standard methodology.

Traceability: The steps to be followed are laid out clearly, making it easy to trace software development through the development lifecycle. This is important for a would-be standard methodology.

30.24.5 PAMELA

The PAMELA methodology impacts the "Lack of an Established Ada Software Development Methodology" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This speeds the coding step of the development process enough to pay for the acquisition costs for some organizations. This greatly increases its chances for adoption as a standard for Ada.

Ease of use: Good tools make PAMELA easier to use, but the basic methodology is not very complicated. This makes PAMELA a stronger candidate of adoption as the standard Ada methodology.

Ease of learning: Training courses are offered frequently. A good book has also been written on this methodology that assists in understanding how it works. This makes PAMELA easier to insert in an organization.

Traceability: PAMELA development follows a well-documented path, making it easy to trace development progress. This is important for a standard methodology.

Auto coding: PAMELA produces Ada code directly from the input design information, eliminating the often expensive step of trying to get a good compilation of Ada code written. If the Ada compiler happens to be slow, this can dramatically improve Ada's appeal for difficult real-time applications.

30.24.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Lack of an Established Ada Development Methodology" problem in the following ways:

Ease of learning: Warnier-Orr is an established methodology that has been in use over ten years, and there is a wealth of documentation available to help learn to use it. Training course availability is no longer adequate, however, which would work against its being adopted as a standard for Ada.

Traceability: Warnier-Orr uses a rigorous procedure, so design traceability using its milestones accurately assesses status. This is an important consideration when looking at a potential standard Ada methodology.

30.24.7 Jackson

The Jackson methodology impacts the "Lack of Established Ada Software Development Methodology" problem in the following ways:

Ease of use: The tools make Jackson easier to use, but the basic methodology is quite complicated. This is even more of a problem for real-time work, which makes it unlikely that Jackson will be adopted as a real-time Ada standard methodology.

Ease of learning: Training courses have been conducted for over ten years and are still offered. Several good books have also been written on this methodology that assist in understanding how it works. This is an important plus for a standard to have.

Traceability: The Jackson process is one of the most rigorous of any methodology, so it is among the easiest to trace development progress. This is important for a standard.

30.25 LACK OF ESTABLISHED ADA SOFTWARE STANDARDS AND GUIDELINES

Sonicraft is among the growing number of software development organizations that measures the productivity of each designer and manager, rewarding them primarily on this measured productivity and the quality of their software products. This has been proven over and over to be a sound practice since we have observed that the difference in productivity between the best and the worst programmers to be consistently about a factor of ten. Besides keeping turnover of the best people very low, this policy also strongly encourages the worst to either quickly improve or to find another line of work.

The reason all organizations don't use this practice is that it takes a considerable amount of effort to make it work right. The measurands we use are, by now, quite standard (operational lines of code and hours charged to the project), with good tools available to automate their collection. The hardest part is to establish standards that can be anchored in some facts, such as the ubiquitous "national average productivity", or a baseline formed from several similar projects done in our environment [Davi85].

Here is the problem that Ada introduces. Because of the difficulties in estimating the number of lines of code or labor hours that a project should take [Problem #23], the credibility of the standard is jeopardized. As long as the standards were based on measured accomplishments of others, they were accepted as a challenge. But if they must be based on theory because Ada has a dearth of real data, their motivational value is greatly reduced.

Compounding the problem is the observation [Problem #23] that the productivity of Ada programmers is likely to start out worse than with other languages, then rapidly improve, ending up with higher productivity than other common languages. This makes productivity data collection far more difficult, since it is continually changing over a period of two or three years. It also makes the data from other organizations more suspect since it is hard to tell exactly where on the learning curve they might be operating.

The following methodology features affect this generic Ada problem:

- o Ada-oriented
- o Methodology tools
- o Problem definition
- o Design quality
- o Auto coding
- o Design consistency

30.25.1 Yourdan

The Yourdan methodology impacts the "Lack of Established Ada Standards and Guidelines" problem in the following ways:

Methodology tools: Yourdan tools are readily available from more than one vendor. The tools are mature, having been released over ten years ago and modified several times since then. This helps stabilize the development process, making productivity easier to estimate.

Problem definition: Yourdan provides a fairly close model of the real world when an application is developed. This assures that small changes in the problem environment will not lead to massive changes in the solution environment. Since rework is very hard to estimate, this helps the process of setting standards and guidelines.

Design quality: The feedback on design quality provided by the Yourdan methodology also reduces unexpected costs due to rework late in the design lifecycle.

30.25.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Lack of Established Ada Standards and Guidelines" problem in the following ways:

Methodology tools: Functional Decomposition tools are general-purpose packages readily available from more than one vendor. The tools are mature. This makes the development process more predictable.

Design quality: The feedback on design quality provided by the Functional Decomposition methodology reduces costs due to rework late in the design lifecycle, making the process more predictable.

Design consistency: The design produced by this methodology can vary over an enormous range because there are no checks provided. This hurts the predictability of the development process and exacerbates the generic Ada problem.

30.25.3 Object Oriented Development

The OOD methodology impacts the "Lack of Established Ada Standards and Guidelines" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This makes the coding step of the development process much more predictable.

Methodology tools: OOD tools are available from just one vendor. The tools are still not mature. This single source helps make the development process more predictable, but the immaturity makes it less predictable.

Problem definition: OOD excels at developing a top-level solution that faithfully models the real world. This improves predictability because small changes in the problem space will come through as small changes in the solution space.

Design quality: The feedback on design quality provided by the OOD methodology reduces uncertainty due to rework late in the design lifecycle.

Design consistency: The design produced by this methodology is quite consistent because there are checks provided. This controls uncertainty due to missed communication in the development group.

30.25.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Lack of Established Ada Standards and Guidelines" problem in the following ways:

Problem definition: The SADT methodology provides symbolic representation that accurately models the real world. This makes solutions more stable, reducing the uncertainties due to massive rework late in the design cycle.

Design quality: The feedback on design quality provided by the SADT methodology reduces uncertainty due to rework late in the design lifecycle.

Design consistency: The design produced by this methodology varies over a limited range because of the checks provided. This controls communication costs.

30.25.5 PAMELA

The PAMELA methodology impacts the "Lack of Established Ada Standards and Guidelines" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This stabilizes the coding step of the development process, increasing the predictability of doing a job in Ada.

Methodology tools: PAMELA tools are available from just one vendor. The tools are still not mature. This lack of competition makes the process easier to estimate, but the immaturity could offset this by driving changes in future (improved) releases.

Design quality: The feedback on design quality provided by the PAMELA methodology reduces uncertainty due to rework late in the design lifecycle.

Auto coding: PAMELA produces Ada code directly from the input design information, eliminating the often unpredictable step of trying to get a good compilation of Ada code written. If the Ada compiler happens to be slow, this can cause major variances in Ada productivity.

Design consistency: The design produced by this methodology is fairly consistent because there are checks provided. This controls communication costs in the development group.

30.25.6 **Warnier-Orr**

The Warnier-Orr methodology impacts the "Lack of Established Ada Standards and Guidelines" problem in the following ways:

Design quality: The feedback on design quality provided by the Warnier-Orr methodology is not as strong as with other methodologies, and uncertainties due to rework late in the design lifecycle could become more problematic.

Design consistency: The design produced by this methodology is restristed to a narrow range because the process is so simple. This reduces uncertainty due to missed communication in the development team.

30.25.7 **Jackson**

The Jackson methodology impacts the "Lack of Established Ada Standards and Guidelines" problem in the following ways:

Methodology tools: Jackson needs are satisfied by general-purpose tools which are mature, having been released over ten years ago and modified several times since then. This reduces the uncertainty in cost and schedule estimations for projects using these tools.

Design quality: The guarantee of design quality provided by the Jackson methodology reduces uncertainties due to rework late in the design lifecycle.

Design Consistency: Jackson is one of the most consistent of the common methodologies, reducing the uncertainties caused by missed communications among the team members.

30.26 PRODUCTIVITY IMPACTS OF ADA

Of the preceding 25 problems, 18-have been shown to influence the cost of an embedded Ada system. On the other hand it is often true that problem identification is the hardest part: once you know what the problem is, the solution is sometimes obvious. This report is therefore not as discouraging as it may seem at first reading.

In the case of Ada, there are some long-term cost benefits that are the ultimate reason it has been adopted as the standard language for Department of Defense embedded mission-critical software [DoD87]. And other languages are not problem-free: MGEN Smith reported that 70 to 80 percent of the late Air Force projects are having software problems [PCRA86]. At the same conference MGEN Salisbury reported another need for a standard language like Ada that can handle a wide variety of applications: The Standard Army Management Information System (STAMIS) had grown to 750 systems, now reduced to 110. Using Ada is expected to cut it to 37 programs.

The one area which is probably the biggest source of Ada productivity problems is the speed of the compiler. Compilation speeds have been steadily dropping for VAX-host, Intel-target compilers. Sonicraft has seen a times ten improvement here in the last two years, with another big improvement reported to be in the offing for a compiler to be validated in September 1987 [Silv87].

For the present, however, Ada does have serious productivity impacts which can price it out of the market for many projects IF YOU LOOK AT DEVELOPMENT COSTS ONLY.

The following methodology features affect this generic Ada problem:

- o Ada-oriented
- o Methodology tools
- o Ease of use
- o Ease of learning
- o Design quality
- o Auto coding
- o Design consistency

30.26.1 Yourdan

The Yourdan methodology impacts the "Productivity Impacts of Ada" problem in the following ways:

Methodology tools: Yourdan tools are readily available from more than one vendor. The tools are mature, having been released over ten years ago and modified several times since then. This competition keeps acquisition costs low, and the maturity keeps development costs low.

Ease of use: The good tools make Yourdan even easier to use, but the basic methodology is not very complicated. This cuts rework costs due to misuse of the methodology.

Ease of learning: Training courses have been conducted by Yourdan for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This cuts training costs.

Design quality: The feedback on design quality provided by the Yourdan methodology reduces costs due to rework late in the design lifecycle.

30.26.2 Functional Decomposition

The Functional Decomposition methodology impacts the "Productivity Impacts of Ada" problem in the following ways:

Methodology tools: Functional Decomposition uses general-purpose tools readily available from more than one vendor. The tools are mature. This competition keeps acquisition costs low, and the maturity keeps development costs low.

Ease of use: Using tools make Functional Decomposition even simpler, but the basic methodology is not very complicated. This cuts rework costs due to misuse of the methodology.

Ease of learning: Training courses have been conducted by several vendors for over ten years and are still offered frequently. Several good books have also been written on this methodology that assist in understanding how it works. This cuts training costs.

Design quality: The feedback on design quality provided by the Functional Decomposition methodology reduces costs due to rework late in the design lifecycle.

Design consistency: The design produced by this methodology can vary over an enormous range because there are no checks provided. This increases communication costs above what they could have been, thereby decreasing productivity.

30.26.3 Object Oriented Development

The OOD methodology impacts the "Productivity Impacts of Ada" problem in the following ways:

Ada-oriented: The OOD methodology contains guidelines for moving from symbology to Ada coding. This speeds the coding step of the development process, increasing productivity in Ada.

Methodology tools: OOD tools are available from just one vendor. The tools are still not mature. This lack of competition drives up acquisition costs, and the immaturity keeps development costs at risk.

Ease of use: The tools make OOD easier to use, but the basic methodology is not very complicated. This cuts rework costs due to misuse of the methodology.

Ease of learning: Training courses are offered frequently. A good book and a sequel have also been written on this methodology that assist in understanding how it works. This cuts training costs.

Design quality: The feedback on design quality provided by the OOD methodology reduces costs due to rework late in the design lifecycle.

Design consistency: The design produced by this methodology is fairly consistent because there are checks provided. This controls communication costs in the development group.

30.26.4 Structured Analysis and Design Technique

The SADT methodology impacts the "Productivity Impacts of Ada" problem in the following ways:

Ease of use: The methodology is fairly complex, driving up the development costs compared to alternative methodologies.

Design quality: The feedback on design quality provided by the SADT methodology reduces costs due to rework late in the design lifecycle.

Design consistency: The design produced by this methodology varies over a limited range because of the checks provided. This controls communication costs.

30.26.5 PAMELA

The PAMELA methodology impacts the "Productivity Impacts of Ada" problem in the following ways:

Ada-oriented: The PAMELA methodology automates the process of moving from symbology to Ada coding. This speeds the coding step of the development process, increasing productivity in Ada.

Methodology tools: PAMELA tools are available from just one vendor. The tools are still not mature. This lack of competition drives up acquisition costs, and the immaturity keeps development costs at risk.

Ease of use: The tools make PAMELA easier to use, but the basic methodology is not very complicated. This cuts rework costs due to misuse of the methodology.

Ease of learning: Training courses are offered frequently. A good book has also been written on this methodology that assists in understanding how it works. This cuts training costs.

Design quality: The feedback on design quality provided by the PAMELA methodology reduces costs due to rework late in the design lifecycle.

Auto coding: PAMELA produces Ada code directly from the input design information, eliminating the often expensive step of trying to get a good compilation of Ada code written. If the Ada compiler happens to be slow, this can drastically improve Ada productivity.

Design consistency: The design produced by this methodology is fairly consistent because there are checks provided. This controls communication costs in the development group.

30.26.6 Warnier-Orr

The Warnier-Orr methodology impacts the "Productivity Impacts of Ada" problem in the following ways:

Ease of learning: Warnier-Orr is an established methodology that has been in use over ten years, and there is a wealth of documentation available to help learn to use it. Training course availability is no longer adequate, however, so training costs could be high until an in-house instructor becomes available.

Design quality: The feedback on design quality provided by the Warnier-Orr methodology is not as strong as with other methodologies, and costs due to rework late in the design lifecycle could occur more frequently.

Design consistency: The design produced by this methodology is restristed to a narrow range because the process is so simple. This reduces communication costs in the development team.

30.26.7 Jackson

The Jackson methodology impacts the "Productivity Impacts of Ada" problem in the following ways:

Methodology tools: Jackson tools are general-purpose charting aids. The tools are mature, with many available ten years ago and modified several times since then.

Ease of use: The tools make Jackson easier to use, but the basic methodology is quite complicated. This increases rework costs due to misuse of the methodology.

Ease of learning: Training courses have been conducted for over ten years and are still offered. Several good books have also been written on this methodology that assist in understanding how it works. This cuts training costs.

Design quality: The guarantee of design quality provided by the Jackson methodology reduces costs due to rework late in the design lifecycle.

Design Consistency: Jackson is one of the most consistent of the common methodologies, cutting communication costs for the team

40. APPENDIX D - IMPACTS OF METHOD FEATURES ON GENERIC ADA PROBLEMS

40.6 PROBLEM 6 - IMPACT OF TASKING OVERHEAD

40.6.4 Method Feature 4 - Overuse Of Tasking

Two interviews reported that having this method feature had an unfavorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of having this method feature is that the impact of tasking overhead is reduced if the method attempts to minimize the use of tasking in the Ada application.

The actual impact of this method feature was reported to occur for the following reason(s):

> Lack of guidelines for the use of Ada tasks - Due to the lack of formal guidelines concerning the effective use of Ada tasks, the system was designed using a large number of tasks. The overhead associated with this large number of tasks reduced system performance and efficiency.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.6.10 Method Feature 10 - Design Quality

Only one interview reported that having this method feature had a favorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of tasking overhead is reduced if the design uses Ada tasks effectively.

The impact of this method feature was reported to occur for the following reason(s):

> Emphasis on reducing Ada overhead - The impact of Ada tasking overhead was reduced by minimizing parameter passing between tasks and by minimizing the use of task rendezvous.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.14 PROBLEM 14 - INABILITY TO PERFORM SECURE ADA PROCESSING

40.14.2 Method Feature 2 - Information Hiding

One interview reported that having this method feature had a favorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the ability to perform secure Ada processing is enhanced if information hiding is used to restrict access to programs and data.

The impact of this method feature was reported to occur for the following reason(s):

> Use of Ada features to support information hiding - The method emphasized the use of built-in Ada language features, such as package specifications and the "with" clause, to restrict access to programs and data within the system. The availability of these built-in features reduced the amount of software that had to be written to enforce the security restrictions.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.14.13 Method Feature 13 - Process State Visibility

One interview reported that having this method feature had an unfavorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of the inability to perform secure Ada processing is reduced by the use of a method which provides strong process visibility. The process visibility helps to ensure that restrictions on access to programs are met.

The impact of this method feature was reported to occur for the following reason(s):

> Verifiability - The process visibility provided by the method allows the developer to determine whether restrictions on access to programs are being met by displaying the flow of program control within the system.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.18 PROBLEM 18 - LACK OF ADA SOFTWARE DEVELOPMENT TOOLS

40.18.5 Method Feature 5 - Method Tools

Only one interview reported that having this method feature had a favorable impact on the generic Ada problem. While five other interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the availability of more development tools increases project productivity by reducing the number of design method tasks that are performed manually. The availability of method tools also improves the consistency of the developed products and assists in the enforcement of project standards.

The impact of this method feature was reported to occur for the following reason(s):

> Lack of available method tools - Overall, the lack of available method tools lowered project productivity because it meant that the majority of method-related development tasks had to be performed manually. One interviewee cited the lack of a requirements traceability tool as an instance of a manual task that desperately needed to be automated.
>
> The lack of available method tools also reduced the uniformity and consistency of the system design because the selected method had to be implemented manually and project standards and guidelines had to be enforced manually.
>
> However, one interviewer used the Yourdon Toolkit to support the Yourdon method in the PC environment. This interviewer felt that the Yourdon tools were more than sufficient to meet project needs.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.21 PROBLEM 21 - LACK OF EXPERIENCED ADA PROGRAMMERS

40.21.3 Method Feature 3 - Ada-Oriented

Three interviews reported that having this method feature had a favorable impact on the generic Ada problem. Two interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the lack of experienced Ada programmers is reduced through the use of an Ada-oriented method which provides guidelines for effective use of the Ada language and reduces the amount of Ada training required.

The impact of this method feature was reported to occur for the following reason(s):

> Mapping design into Ada constructs - The ability of the method to map the design into Ada constructs improved project productivity by providing guidelines for performing the mapping, which reduced the effort required to perform this mapping. Design quality was also improved because the method provided guidelines for effective use of the Ada language.
>
> Compatibility with Ada software engineering principles - The compatibility of the method with Ada software engineering principles reduced learning curve time for the method because of the reduced effort required to implement the Ada design.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.21.5 Method Feature 5 - Method Tools

One interview reported that having this method feature had a favorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of the lack of experienced Ada programmers is reduced by the availability of method tools which improve productivity and enhance effective use of the method.

The impact of this method feature was reported to occur for the following reason(s):

Availability of method tools - The availability of method tools improved productivity by automating a number of manual development activities.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.21.6 Method Feature 6 - Ease Of Use

Two interviews reported that having this method feature had a favorable impact on the generic Ada problem. Three other interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the lack of experienced Ada programmers is reduced by a method that is easy to use. The reduction in impact is due to the improved productivity which results from effective use of the method.

The impact of this method feature was reported to occur for the following reason(s):

Maturity of the method - The maturity of the method affected project productivity because the mature methods tended to be easier to use (available tools, effective symbology and terminology, etc..) than the immature methods.

Complexity of method - The complexity of the method affected project productivity because the complex methods were more difficult to use than the simpler methods.

Lack of a formal method - The lack of a formal method affected project productivity because the lack of guidelines for method use made it more difficult to use the method effectively than in cases where comprehensive guidelines exist.

Use of a well-known method - The use of a well-known method tended to improve project productivity because of the higher probability that some of the project personnel were already familiar with the method and knew how to use it effectively.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.21.7 Method Feature 7 - Ease Of Learning

Three interviews reported that having this method feature had a favorable impact on the generic Ada problem. Two interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the lack of experienced Ada programmers is reduced by the use of a method which is easy to learn. The reduction in impact is due to a reduction in the length of the project learning curve.

The impact of this method feature was reported to occur for the following reason(s):

> Lack of formal training - The unavailability of formal training or the decision to forego formal training tended to increase project learning curve because the project personnel required more time to learn how to use the method effectively.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.22 PROBLEM 22 - EXTENSIVE ADA TRAINING REQUIREMENTS

40.22.5 Method Feature 5 - Method Tools

One interview reported that having this method feature had a favorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of the extensive Ada training requirements is reduced by the availability of method tools which can be used as training aids to reduce the required training effort.

The impact of this method feature was reported to occur for the following reason(s):

> Availability of method tools - The availability of method tools reduced project learning curve by providing a software training package for self-paced training and providing method tools to support informal and self-paced training on the project.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.22.6 Method Feature 6 - Ease Of Use

Six interviews reported that having this method feature had a favorable impact on the generic Ada problem. Two interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the extensive Ada training requirements is reduced by the selection of a method that is easy to use, which reduces the Ada training required to ensure effective use of the method.

The impact of this method feature was reported to occur for the following reason(s):

> Complexity of method - The complexity of the method affected learning curve because it was more difficult to use the complex methods effectively than it was to use simpler methods.
>
> Transition from functional to object-oriented approach - The learning curve was increased due to the difficulty in using an object-oriented design method as opposed to a functionally-oriented design method.

Emphasis on abstraction - The learning curve was increased due to the emphasis placed on enforcing abstraction in the early stages of design, not allowing the developers to provide the lower-level details too early in the design process. Some of the developers seemed to have difficulty in implementing this approach.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.22.7 Method Feature 7 - Ease Of Learning

Three interviews reported that having this method feature had a favorable impact on the generic Ada problem. Four interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the extensive Ada training requirements is reduced because the use of a method that is easy to learn reduces the project learning curve.

The impact of this method feature was reported to occur for the following reason(s):

Complexity of method - The complexity of the method affected project learning curve because it was more difficult to learn to use the complex methods effectively than it was to learn the simpler methods.

Use of a well-known method - The use of a well-known method tended to reduce project learning curve because more of the project personnel were already familiar with the well-known methods (thus requiring less training in their usage) than with the lesser-known methods.

Transition from functional to object-oriented approach - The learning curve was increased due to the difficulty in understanding the principles involved in making a transition from a functionally-oriented design method to an object-oriented design method.

Immaturity of method - The learning curve was increased due to the lack of available documentation concerning process abstraction-oriented design methods.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.24 PROBLEM 24 - LACK OF ESTABLISHED ADA SOFTWARE DEVELOPMENT METHOD

40.24.3 Method Feature 3 - Ada-Oriented

Five interviews reported that having this method feature had a favorable impact on the generic Ada problem. Three other interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the lack of an established Ada software method is reduced by the use of an Ada-oriented method because the guidelines that are provided for mapping the design into the Ada language enhance the ability of the developers to use Ada effectively.

The impact of this method feature was reported to occur for the following reason(s):

> Mapping design into Ada constructs - The ability of the method to map the design into Ada constructs affected project productivity because less effort was required to perform this mapping for the Ada-oriented method than for the non-Ada-oriented methods due to the guidelines provided by the method.
>
> Compatibility with Ada software engineering principles - The compatibility of the method with Ada software engineering principles affected overall design quality because the Ada-oriented methods provided more guidance for the development of an efficient Ada design than did the non-Ada-oriented methods.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.24.5 Method Feature 5 - Method Tools

One interview reported that having this method feature had a favorable impact on the generic Ada problem. Five interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the lack of an established Ada software method is reduced due to the availability of method tools which help to provide a more consistent implementation of the method.

The impact of this method feature was reported to occur for the following reason(s):

> Design consistency - Due to the ability of method tools to support design standardization, the consistency of the

overall design was higher for those projects which used method tools than for those projects which used no method tools.

Guidance for use of method - The projects which used method tools received more guidance for use of their methods than those projects which used no method tools due to the guidelines for method use that were provided by the tools.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.24.6 **Method Feature 6 - Ease Of Use**

Three interviews reported that having this method feature had a favorable impact on the generic Ada problem. Three interviews also reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the lack of an established Ada software method is reduced for a method which is easy to use because less effort is required to use the method effectively.

The impact of this method feature was reported to occur for the following reason(s):

Maturity of the method - The maturity of the method affected project development because the mature methods tended to be easier to use (available tools, effective symbology and terminology, etc..) than the immature methods.

Lack of a formal method - The lack of a formal method affected project development because the lack of a framework (guidelines) for method use made it more difficult to use the method effectively than in cases where a framework exists.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.24.11 **Method Feature 11 - Traceability**

One interview reported that having this method feature had a favorable impact on the generic Ada problem. Four interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the lack of an established Ada software method is reduced by the use of a method which requires traceability between the outputs and activities of the various method phases. This traceability provides an increased capability to evaluate the effectiveness with which the method is used.

The impact of this method feature was reported to occur for the following reason(s):

Verification/validation of method - The lack of traceability of the methods affected project development by providing minimal guidelines to ensure that the products generated were complete and accurate and that all activities in previous project life cycle phases were completed before proceeding to the next phase.

Lack of requirements traceability - The lack of requirements traceability support provided by the methods reduced the completeness and correctness of the generated products by providing minimal support to ensure that all of the project requirements have been met.

Availability of automated traceability tools - The availability of automated traceability tools increased the completeness and correctness of the generated products by providing an automated means to ensure that project requirements have been met.

Transition from functional to object-oriented approach - Project traceability was decreased due to the difficulty in maintaining design traceability while making a transition from a functionally-oriented design method to an object-oriented design method.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.24.13 Method Feature 13 - Process State

One interview reported that naving this method feature had an unfavorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of the lack of an established Ada software method is reduced due to the use of a method which can accurately model process states and transitions. This feature provides the developer with a means to model the behavior of the process and is critical for real-time weapons system applications.

The impact of this method feature was reported to occur for the following reason(s):

Inability to represent process states and transitions - The inability of the method to represent process states and transitions reduces the corfectness of the system design by not providing a means to evaluate the real-time behavior of the system processes.

In one case, the developers enhanced their selected method by adding symbology and terminology which provided a means to represent process states, transitions, conditions, and actions.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.24.15 **Method Feature 15 - Design Consistency**

Two interviews reported that having this method feature had an unfavorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of the lack of an established Ada software method is reduced due to the capability of the method to encourage design uniformity and consistency across the project.

The impact of this method feature was reported to occur for the following reason(s):

Lack of guidelines for method usage - The lack of guidelines for use of the method reduces the consistency of the design by allowing the members of the development team to use their own design approaches rather than one which was established for the project as a whole.

One interviewee cited two specific instances where the lack of guidelines was significant on their project. The first instance was the ambiguity that was allowed in the problem definition. The second was the lack of guidance provided in the selection of objects and operations.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.25 PROBLEM 25 - LACK OF ESTABLISHED ADA SOFTWARE STANDARDS AND GUIDELINES

40.25.3 Method Feature 3 - Ada-Oriented

Three interviews reported that having this method feature had a favorable impact on the generic Ada problem. Six interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is in-conclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of the lack of established software standards and guidelines is reduced because of the standards that are provided by an Ada-oriented method for implementation of the design in the Ada language.

The impact of this method feature was reported to occur for the following reason(s):

> Compatibility with Ada software engineering principles - The compatibility of the method with Ada software engineering principles affected overall design quality because the Ada-oriented methods provided more standards for the design and implementation of Ada software than did the non-Ada-oriented methods.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.25.6 Method Feature 6 - Ease Of Use

Two interviews reported that having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the lack of established software standards and guidelines is increased if the standards that are used on the project do not provide the developers with guidance concerning the effective use of the Ada language.

The impact of this method feature was reported to occur for the following reason(s):

> Inadequate software standards - The use of inadequate software standards affected overall design quality because the developers were not provided with enough guidance concerning the effective use of the Ada language for design and implementation.

Some developers elected to enhance the the software standards that were used on their projects by creating additional project standards or by supplementing their specified project standards through the use of in-house development standards.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.26 PROBLEM 26 - PRODUCTIVITY IMPACTS OF ADA

40.26.1 Method Feature 1 - Process Visibility

Only one interview reported that having this method feature had an unfavorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of the productivity impacts of Ada is reduced by the use of a design method which provides process visibility. This visibility enables the developers to assess and evaluate the interactions between system processes.

The impact of this method feature was reported to occur for the following reason(s):

> Impact of design errors - The inability of the method to provide adequate process visibility resulted in a reduction in project productivity due to extensive design rework. Due to the lack of process visibility, the developers were not able to accurately determine system behavior; thus, a number of system deficiencies were not discovered until the implementation phase. The effort required to correct these deficiencies at the implementation phase is much greater than that required in the earlier project stages.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.26.3 Method Feature 3 - Ada-Oriented

Two interviews reported that having this method feature had a favorable impact on the generic Ada problem. Only one interview reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the productivity impacts of Ada is reduced by the use of an Ada-oriented method which provides guidance in the mapping of the design into Ada. This guidance decreases the effort required to perform this activity.

The impact of this method feature was reported to occur for the following reason(s):

> Mapping design into Ada constructs - The ability of the method to map the design into Ada constructs affected project productivity because less effort was required to perform this mapping for the Ada-oriented method than for

the non-Ada-oriented methods due to the guidelines provided by the method.

Complexity of method - The complexity of the method affected project productivity because it was more difficult to learn the complex methods and to use them effectively than it was for the simpler methods.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.26.5 Method Feature 5 - Method Tools

Two interviews reported that having this method feature had a favorable impact on the generic Ada problem. Four interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the productivity impacts of Ada is reduced by the availability of method tools which improve productivity by automating tasks that would otherwise be performed manually.

The impact of this method feature was reported to occur for the following reason(s):

Availability of method tools - The projects which were able to use method tools experienced an increase in productivity over those which had no method tools. This was due to the ability of the tools to automate a number of the manual method development activities.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.26.6 Method Feature 6 - Ease Of Use

Two interviews reported that having this method feature had a favorable impact on the generic Ada problem. Two interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the productivity impacts of Ada is reduced when the selected method is easy to use because less effort is the required to learn how to use the method effectively.

The impact of this method feature was reported to occur for the following reason(s):

Effective use of the method - The projects that had guidelines for use of the methods realized higher productivity than those projects which had no guidelines because the existence of these guidelines reduced the effort required to use the method effectively.

Effective use of the Ada language - The projects that had guidelines for use of the Ada language realized higher productivity than those projects which had no guidelines because the existence of these guidelines reduced the effort required to effectively use the Ada language.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.26.7 Method Feature 7 - Ease Of Learning

Three interviews reported that having this method feature had a favorable impact on the generic Ada problem. Two interviews reported that not having this method feature had an unfavorable impact on the generic Ada problem.

The expected impact of this method feature is that the impact of the productivity impacts of Ada is reduced when using a method that is easy to learn. This ease of learning reduces the project learning curve.

The impact of this method feature was reported to occur for the following reason(s):

Complexity of method - The complexity of the method affected productivity because the project learning curve was longer when learning to use the complex methods than when learning to use the simpler methods.

Difficulty in understanding OOD and Ada - The difficulty in understanding the implementation of an object-oriented design in Ada resulted in reduced project productivity due to design rework.

Use of a well-known method - The use of a well-known method affected productivity because more of the project personnel were already familiar with the well-known methods (thus reducing the project learning curve) than with the lesser-known methods.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.

40.26.10 Method Feature 10 - Design Quality

Two interviews reported that having this method feature had a favorable impact on the generic Ada problem. The information provided concerning the impact of this method feature is inconclusive, because the number of interviewees that identified the impact of this feature is less than three (the minimum required for statistical significant's).

The expected impact of this method feature is that the impact of the productivity impacts of Ada is reduced when using a method which produces a high-quality design. The improvement in productivity is due to reduction in the amount of design rework that has to be performed. Productivity is also realized due to increased testability, maintainability, portability, and reusability.

The impact of this method feature was reported to occur for the following reason(s):

> Reduction in rework - The high-quality design produced by the methods increased project productivity by reducing the amount of rework (redesign, optimization, correction of deficiencies) required for the project.
>
> Reduced testing effort - The high-quality design produced by the methods increased project productivity because the modularity and localization of the design reduced the effort required to perform software and system testing.

The expected impact of the method feature on the generic Ada problem agrees with the actual impact as reported from the interviews.